Shorthand | Coursera

coursera.org/learn/programming-fundamentals/supplement/9khxH/shorthand

Shorthand

Shorthand

C (and many other programming languages) has shorthand—also called *syntactic sugar*—for a variety of common operations. These shorthands do not introduce any new behaviors. Instead, they just provide a shorter way to write common patterns of existing things we have seen. This table shows the most common shorthand notations in C:

These shorthands have exactly the same effect as their expanded meanings. Consequently, when you encounter a shorthand statement while executing code, you can execute it by considering what its fully written out form is, and performing the effects of that statement.

Shorthand	Meaning
`x += y;`	`x = x + y;`
`x -= y;`	`x = x - y;`
`x *= y;`	`x = x * y;`
`x /= y;`	`x = x / y;`
`x++;`	`x = x + 1;`
`++x;`	`x = x + 1;`
`x--;`	`x = x - 1;`
`--x;`	`x = x - 1;`

For the shorthands that combine an operation with assignment (such as *=), if the right hand side involves other operation, you should treat the right hand side as parenthesized when you expand the shorthand. That is

x *= y+3;

behaves just like

x = x * (y +3);

This is because the intent is to multiply the left side (x) by the right side (y+3) and then assign that to x.

Another possible shorthand is to omit the curly braces around single statement blocks of code in certain cases. For example, if the “then” and/or “else” clause of an if statement is one single statement, the curly braces are not required. While you may encounter code written this way by other people, it is highly inadvisable to write code this way. Omitting the curly braces presents a danger if you modify the code in the future. If you add another statement to the clause, but forget to add curly braces, then the statement will not actually be part of the clause, but rather the first statement after the **if**. Such errors have produced high-profile security vulnerabilities recently.