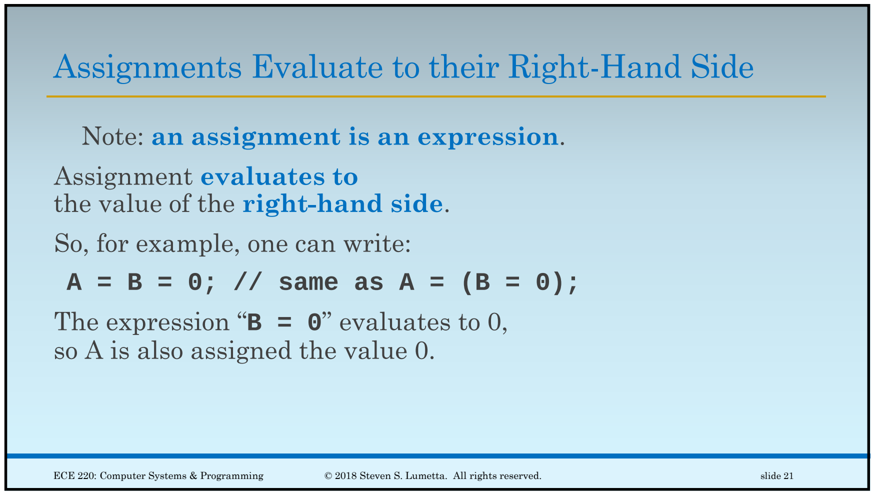


Assignments Evaluate to their Right-Hand Side

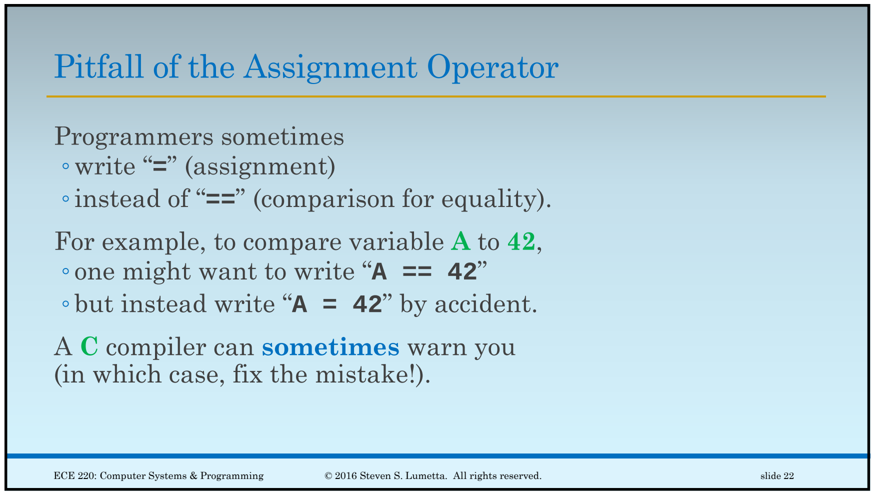
Note: **an assignment is an expression**.

Assignment **evaluates to** the value of the **right-hand side**.

So, for example, one can write:

```
A = B = 0; // same as A = (B = 0);
```

The expression "**`B = 0`**" evaluates to 0, so A is also assigned the value 0.

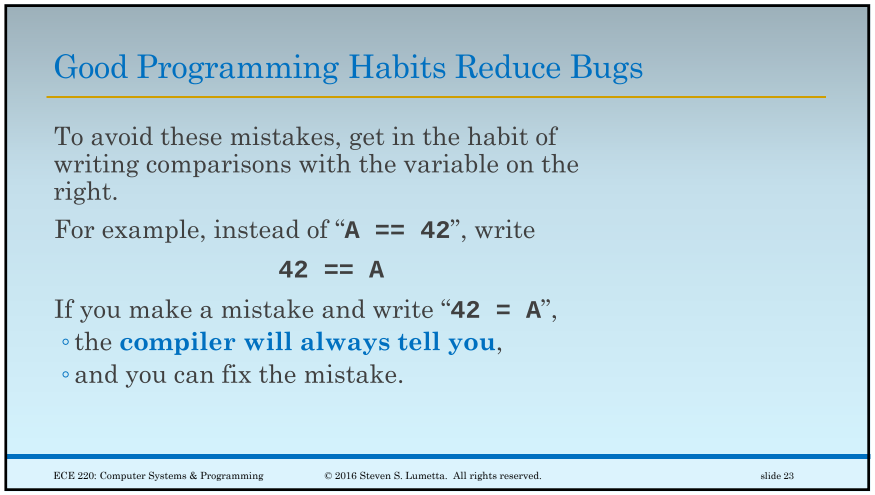


Pitfall of the Assignment Operator

Programmers sometimes
- write "**`=`**" (assignment)
- instead of "**`==`**" (comparison for equality).

For example, to compare variable **A** to **42**,
- one might want to write "**`A == 42`**"
- but instead write "**`A = 42`**" by accident.

A **C** compiler can **sometimes** warn you (in which case, fix the mistake!).



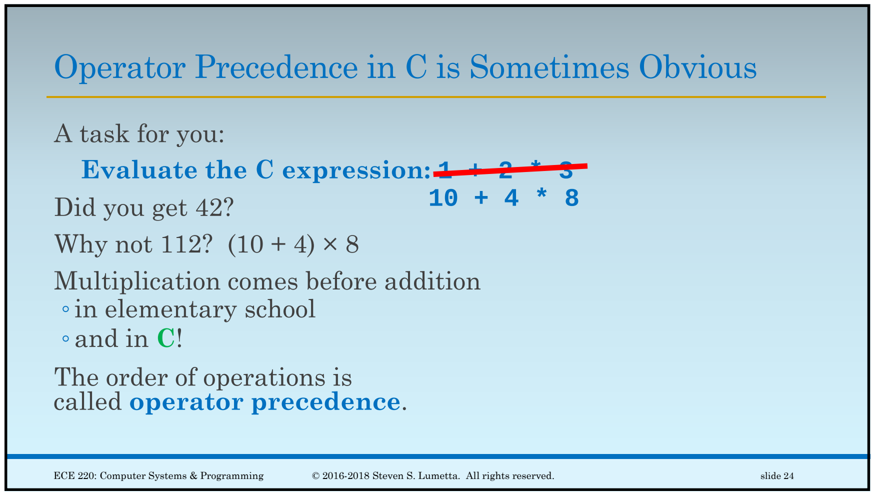
Good Programming Habits Reduce Bugs

To avoid these mistakes, get in the habit of writing comparisons with the variable on the right.

For example, instead of "**`A == 42`**", write

```
42 == A
```

If you make a mistake and write "**`42 = A`**",
- the **compiler will always tell you**,
- and you can fix the mistake.



Operator Precedence in C is Sometimes Obvious

A task for you:

Evaluate the C expression: ~~`1 + 2 * 3`~~
`10 + 4 * 8`

Did you get 42?

Why not 112? $(10 + 4) \times 8$

Multiplication comes before addition
- in elementary school
- and in **C**!

The order of operations is called **operator precedence**.