Assignment

The assignment operator copies the value on its right, and stores the copy inside the **already** existing variable on its left. Here are some examples:

```
1  int sides = 7;   // initialization (not assignment)
2  . . .
3  sides = 10;       // non-range-checked assignment
4  sides = {3.5};    // range-checked assignment (error)
```

- Line 1 is **initialization**; it **may** appear **outside** of a function.
- Lines 3 and 4 are **assignment**; they copy a value into an existing variable.
- All assignment statements **must appear inside a function**.
- **List-assignment,** with the value enclosed in braces, allows the compiler to perform additional range or type checking. Line 4 produces a compiler error because `3.5` cannot be converted to an `int` without loss of information.

Lvalues and Rvalues

An ***lvalue*** is an object that has an address. Such objects can appear on the left-hand-side of an assignment operation. (The "el" stands for "left".) An ***rvalue*** is any value which may appear on the right-hand-side of an assignment.

- A variable may be used as **either** an ***lvalue*** or an ***rvalue***.
- Literals (such as the number `10`, or the string literal `"hello"`), as well as temporaries (such as the value produced by an expression, such as `(3 + 4)`) may only be ***rvalues***.
- Constants and arrays are called **non-modifiable** ***lvalues*** since their names appear on the left side of the assignment operator when they are defined, but cannot be modified later.