use(looper) = morf;

For nonreference return values, this assignment is a syntax error, but it's okay for use(). This is the order of events. First, the use() function is evaluated. That means looper is passed by reference to use(). As usual, the function displays two members and increments the used member to 4. Then, the function returns the reference. Because the return value refers to looper, this makes the final step equivalent to the following:

looper = morf;

C++ allows you to assign one structure to another, so this copies the contents of the morf structure into looper, as is shown when displaying looper.name produces Morf's name and not Looper's. In short, the statement

use(looper) = morf;  // return value a reference to looper

is equivalent to the following:

use(looper);
looper = morf;

Remember

You can assign a value (including a structure or a class object) to a C++ function only if the function returns a reference to a variable or, more generally, to a data object. In that case, the value is assigned to the referred-to variable or data object.

You may have noticed that assigning to a reference return value produces confusing-looking code. The main reason for returning a reference is efficiency, not to enable this peculiar assignment mechanism. However, the assignment property occasionally is handy for some forms of operator redefinition. You can use it, for example, to redefine the array subscript operator [] for a class that defines a string or a more powerful version of an array.

Considerations When Returning a Reference or a Pointer