Do Not Duplicate

"A place for everything and everything in its place." Certainly, I'm not the only person who was given this advice during the formative years. Its initial application was part of my parents' efforts in getting me to clean my room, but I have found it handy in the workplace as well. C++ calls it the "One Definition Rule (ODR)," and contains language like the following:

"No translation unit shall contain more than one definition of any variable, function, class type, enumeration type or template… Every program shall contain exactly one definition of every non-inline function or object that is used in that program."

If compilers can get confused with duplicate definitions, so can you and I. Even mathematicians like to avoid duplication. In algebra, they call it *factoring*. The second line below, a factored version of the first, is not only more readable, but more efficient, since it has fewer operations.

$$\frac{n(2n-1)(2n+1)}{3}+(2n+1)^2 =$$
$$\frac{(n+1)(2n+1)(2n+3)}{3}$$

In programming, we call it *refactoring*. Although the example is now trite, humor me by considering how to write a simple string class. You may first implement the constructor and destructor something like this.

```
class String {
   char* data;
public:
   String(const char* str = "") {
      data = new char[strlen(str) + 1];
      strcpy(data,str);
   }
   ~String() {
      delete [] data;
   }
};
```

You then decide you want copy semantics and therefore define a copy constructor and copy-assignment operator:

```
String(const String& s) {
   data = new char[strlen(s.data) + 1];
   strcpy(data, s.data);
}
String& operator=(const String& s) {
   if (this != &s) {
      char* newData = new char[strlen(s.data) + 1];
      strcpy(newData, s.data);
      delete [] data;
      data = newData;
   }
   return *this;
}
```

There are now three "definitions" of how to create a string object instead of one. While this example is trivial, you get the idea. When you duplicate operations, you create a maintenance nightmare when those operations must be updated. It's better to refactor the common operation into its own function and call that function where it is needed. Just as databases should be normalized so that data is not replicated throughout multiple tables, code should be normalized by adhering to the ODR concept.

Hunt and Thomas[1] call it the DRY principle (Don't Repeat Yourself) and remind us that code repetition comes in many forms. It can happen implicitly like in the string example above, or you can do it on purpose with your editor's cut-and-paste facility (the infamous clone-and-mutate approach to "reuse"). An even more subtle folly is the practice of writing comments that mirror code. When a line of code changes, so must its companion comment – another maintenance headache! In general, comments should describe program behavior at a higher level than code, anyway, and should not just repeat what each line does. Better yet, generate one from the other!

The list goes on. There are more ways to duplicate code than this forum has space to describe, so let me conclude by begging your indulgence as I repeat myself one last time: *don't repeat yourself*. Don't make me tell you again.

[1] *The Pragmatic Programmer*, Addison-Wesley, 2000.