V502 Overloaded Operators I

Bryan Beus

What are the risks of using friend?

Using `friend` breaks encapsulation. We should always be asking, "Is there a better way?" Even if the answer is "no," we should still ask.

Allowing external members to define private data can break a Class's integrity.

We can only grant friend status. We cannot retract that status, once given.

Why is it important to use appropriate const correctness?

Const protects data that should not be changed from changing.

In general, everything that can be made const within the intentions of the software, should be made const. This prevents developers from designing ineffective code.

When should you return by value versus return by reference?

Regarding overloaded operators, we can choose to return by reference when we expect the operator to be called immediately after the current operation.

For example:

```
cout << Foo << endl;
```

The first pair, `cout << Foo` calls the overloaded operator, `operator<<`, performs a function, and then returns the `ostream` itself, thus allowing for `(ostream) << endl;` to perform the function again.

Outside of overloaded operators, we generally return by value. Returning by reference runs the risk of returning a variable that was created within the scope of the function returning the object, and once this function goes out of scope, so does the referenced data. Therefore, returning by value is more common.

We can, still, return by reference, even in this situation. For example, if the object that is being returned was created outside the scope of the returning function, returning by reference to that object might not have adverse effects. But these are rare scenarios.