V503 – Overloaded Operators II

Bryan Beus

Why should the parameter for Foo::operator+ be a reference to const or pass by value?

Overloaded arithmetic operators should be free functions, because the member function version cannot do conversions of the first parameter.

Why should you write operators in terms of other operators?

This allows us to avoid repeating ourselves. We set an initial overloaded operator that performs a preliminary function, and then have the second overloaded operator complete the function. This allows for overloaded operators that can perform in either the simpler of the operators (`+`) or more complex (`+=`) form.

What is the difference in syntax between a global overloaded operator and a member overloaded operator?

A global overloaded operator:

```
ostream& operator<< (ostream& os, int myData) {
    return &os;
}
```

A member overloaded operator:

```
Foo& Foo::operator<< (const Foo& foo, int myData) {
    return *this;
};
```