Doc No: X3J16/91-0139
WG21/N0072
Date: 20 November 1991
Reply to: Dag Brück

Operator Overloading on Enumerations

Dag M. Brück

Department of Automatic Control
Lund Institute of Technology
Box 118, S-221 00 Lund, Sweden

The working draft, Section 13.4, says:

- An operator function must either be a member function or take at least one argument of a class or a reference to a class.

The Annotated C++ Reference Manual offers some explanation on page 330:

- This implies that the meaning of operators applied to nonclass types cannot be redefined. The intent is to make C++ extensible, not mutable.

I propose a slight change in the restriction of argument types:

- An operator function must either be a member function or take at least one argument of a class, a reference to a class, an enumeration, or a reference to an enumeration.

Originally, enumerations were just integers in disguise; in particular, implicit conversion from `int` to enumerations was allowed. Overloading operators based on enumerations could then potentially redefine operations on built-in types.

The current language definition no longer allows implicit conversion from `int` to enumeration, and enumerations are not even integral types anymore. Consequently, there is no longer any risk that a overloading of, for example,

```
enum Status { bad, worse, terrible };
Status operator & (Status, Status);
```

redefines operations on built-in types. This proposal does not allow user-defined assignment of enumerations because the assignment operator must be a member function.

Allowing operator overloading on enumerations is a useful, natural, and as far as I can tell, safe generalization of C++. It does not break any existing code.