Date: 15 February 1997
To: X3J3
From: Van Snyder
Subject: Separate module specification/implementation

The capability advocated in Section 23 of X3J3/97-114r2, that allows separation of module specification and implementation, was added to the "C" list during X3J3 meeting 140. Illustrative editorial changes that apply to ISO/IEC 1539:1991(E) were suggested in X3J3/97-114r2, but are not reproduced here.

Section 23 of 97-114r2: Module Interfaces and Implementations Should Be Separated

Both Modula and Ada allow separating module interface and implementation into separate program units. There are several advantages of this approach, as compared to having interface and implementation in a single program unit.

- If one changes the implementation but not the interface, one does not trigger recompilation of dependent modules.
- Vendors of software component libraries who do not wish to publish the source form of the implementation of their product could publish machine-readable source form of the interface without compromising their "trade secrets."
- Several Ada implementors have found that a compiler that reads the source form of the interface is just as fast as a compiler that reads a "pre-compiled" form of the interface. This simplifies the compiler, and further reduces the opportunities for "compilation cascades." (In systems that keep a pre-compiled form of interface modules, compiling the interface module, even if it's not changed, can trigger a "compilation cascade.")
- Ada-95 uses the separation of interface and implementation, together with *private* specifications in the interface module, to provide the analog of C++ *public*, *private* and *protected* visibility control.

- One can construct limited mutual dependencies between modules: Implementation modules A and B can each USE the other's interface module (but recursive dependence between interface modules is prohibited).

- It provides a natural mechanism for truly private procedures – just don't mention them in the interface module.

Fortran could introduce separation of interface and implementation in a way that is compatible to existing software. Modules that begin with distinguished statements, e.g. INTERFACE MODULE and IMPLEMENTATION MODULE would be just that. An INTERFACE module contains an *interface-body* for every procedure it wishes to publish (but not in an interface block, so it's a module procedure); the full procedure must be declared in the implementation module, with the same characteristics . An implementation module automatically incorporates its corresponding interface module. USE statements refer only to interface modules, or undistinguished modules, never to implementation modules.

Same dummy argument names?

This would not be a large change in the standard document, and would be almost completely independent from all other features of the language.