****
J3/02-296r2

Date: November 2002
To: J3
From: Aleksandar Donev
Subject: Reinstating Deferred Bindings
Reference: J3-007R3, J3/02-271r1
****

Summary

I propose to do the work necessary to bring back deferred bindings into F2x, and the specific syntax arrived at during a discussion with Kurt and Van.

J3 response:

J3 will recommend that deferred bindings be added to the draft standard, using the proposed specifications as a design guide.

Motivation

We want three useful debugging/diagnostic tools for deferred bindings:
1. They must be part of the syntax so that from the declaration of an extensible type it is clear that a particular type-bound procedure is deferred.
2. They must provide compile-time checks when extending the type, forcing the user to do something with the deferred procedure (either defer it again or override)
3. They should provide a runtime debugging protection/traceback if the deferred procedure is envoked.

Solution

We allow the following two forms of deferred binding:
a)
PROCEDURE, DEFERRED :: proc=>my_error_handler
The deferred attribute here means that upon extension another deferral or explicit override is required. This is a compile time check to the programmer.
b)
PROCEDURE, DEFERRED :: proc=>abstract_interface
In this case the compiler will automatically provide a *real* procedure with the specified interface which will do some kind of error-handling, and we say that invocation of this procedure will terminate execution (like invalid IO statements). This is a runtime aid to the programmer.

With this scheme deferred type-bound procedures are treated exactly like regular type-bound procedures, so the standard is simpler and there is no runtime segfaults. The same syntax applies to type-bound generics. There some of the specifics can be explicitly provided by the user, and others can be default debugging ones provided by the compiler.