Subject: Internal subprograms as actual arguments and procedure pointer targets
From: Van Snyder
Reference: 03-258r1, section 1.7

1 Number

TBD

2 Title

Internal subprograms as actual arguments and procedure pointer targets.

3 Submitted By

J3

4 Status

For consideration.

5 Basic Functionality

Allow internal subprograms as actual arguments and procedure pointer targets.

6 Rationale

In many cases where one uses a subprogram as an actual argument, it needs access to entities of which the procedure to which it is passed is unaware. If the actual argument were an internal subprogram, it could access these extra entities by host association.

If the 2003 standard does not require the TARGET attribute for a nonpointer dummy procedure that is a procedure pointer target, we cannot simultaneously allow internal procedures to be actual arguments and prohibit them to be procedure pointer targets.

7 Estimated Impact

Small. Minor changes necessary in Sections 7, 12 and 16, and Annex C.

8 Detailed Specification

8.1 Allow an internal subprogram to be an actual argument

There are two possibilies concerning the host of the internal subprogram:

- Allow the host of the internal subprogram to be recursive. Make it clear that accesses by host association from the internal subprogram to entities of its host are accesses to entities in the instance of its host as of the instant the internal subprogram was used as an actual argument, not to entities in the instance of its host as of the instant the internal subprogram is invoked.

  If the internal procedure cannot be passed to a recursive invocation of its host (which can sometimes be verified by analyzing the host's dummy arguments) there can be no difference between these instances. Nonetheless, there can still be more than one instance of the host. Therefore, procedure arguments would need to have the extra overhead of a pointer to the appropriate instance of the host of the actual argument — even if the host is nonrecursive, because the called procedure cannot efficiently know whether the host of an actual argument associated with a dummy procedure is recursive. Since a procedure that invokes a dummy procedure cannot efficiently know whether the associated actual procedure is or is not an internal procedure, there is extra overhead associated with every procedure reference by way of a dummy procedure. Since a procedure cannot efficiently know whether it is invoked by way of a dummy procedure, there is extra overhead associated with every procedure refrence — even those having nothing to do with internal subprograms or dummy procedures.

There is also extra overhead at every reference from an internal subprogram to its host by host association if the host is recursive: It needs to know to which instance of its host it is referring. Since the internal procedure cannot know whether it is being invoked directly or by way of a dummy procedure, the specification of the instance of the host needs to be provided by the procedure invocation, not by some other data structure (else one could get the wrong instance).

- Require that the host of the internal subprogram is not recursive. In this case, there can only be one instance of the procedure defined by the host subprogram, so a pointer to the instance does not need to accompany the argument. This restriction could later be relaxed.

8.2 Allow an internal subprogram to be a procedure pointer target

Similar considerations regarding recursive hosts apply. It is unavoidable that the same restriction applies to internal subprograms used as procedure pointer targets and used as actual arguments: A procedure that has a dummy procedure cannot efficiently know whether the actual argument is an internal procedure or a procedure pointer associated with an internal procedure, and an internal subprogram cannot efficiently know whether it is invoked directly, by way of a procedure pointer, or by way of a dummy procedure.

If internal subprograms of recursive hosts are allowed to be procedure pointers, make it clear that the instance of the host to which accesses from the internal subprogram to that host by host association refer, when it is invoked by way of the pointer, is the instance as of the instant the procedure was associated with the pointer, not the instance as of the instant the internal subprogram is invoked via the pointer.

Make sure that procedure pointers associated with an internal subprogram become undefined when the instance of the procedure defined by its host subprogram that was in existance at the instant the pointer association was established ceases to exist.

9 History