Subject: Minor correction in 16.4.1.3 Host association
From: Van Snyder

1 Introduction

Richard Maine has pointed out that the list in 16.4.1.3 [411:14-31] claims that an external function is a local identifier, when in fact it is a global identifier. Richard and I and others have remarked that it doesn't seem appropriate for the list to give the appearance of defining what is and is not a local identifier anyway, since that's already done in 16.2 [406:3-11].

2 Edits

Edits refer to 04-007. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

[Editor: at the end of the second paragraph of 16.4.1.3 replace "A name that appears" by "If an identifier appears".] 411:13

[Editor: in the first phrase after the list in 16.4.1.3 delete "is a local identifier in the scoping unit and".] 411:32

[Editor: in the first phrase after the list in 16.4.1.3 replace "name" by "identifier" twice.] 411:32-33

[Editor: Replace the second sentence following the list in 16.4.1.3 by the following.] 411:33-36

If a scoping unit is the host of a derived-type definition or a subprogram, any entity of the host that has the same nongeneric identifier as the derived-type definition or subprogram is inaccessible by that identifier by host association.

3 Questions

16.4.1.3 appears to allow a local entity of a scoping unit that is not accessed by use association to have a name that is the same as a generic identifier of the host of that scoping unit, and to allow that the generic entity of the host is nonetheless accessible within the scoping unit by that identifier by host association. This is clearly absurd unless the local identifier is a generic identifier.

It is possible for a function to overload a structure constructor. The derived type definition from which the structure constructor arises does not have a generic name. So if a type is defined in the host scoping unit, and the function is defined in a contained scoping unit, does the function overload the structure constructor, or override it?

Is more repair required in 16.4.1.3 than is so far contemplated here? If so must it be done by the interp mechanism?