Comments on Section 16

To: J3
From: Craig Dedo
Date: August 13, 2002
Subject: Comments on Section 16

1 Edits

394:6-7 [At [63:2], we see that an enumeration is a type alias, and an enumerator is a named constant. The list at [394:4-7] has named constants, type aliases and enumerations, but not enumerators. For consistency, either enumerators need to be in the list, or enumerations don't need to be. Choose the latter for simplicity. Editor: Delete "enumerations,".]

394:23, 27 [It's unlikely that one name is used to identify another. Editor: Delete "the name of" twice.]

395:5 [It's unlikely that one name is used to identify another. Editor: Delete "the name of".]

396:bottom [Editor: Delete Note 16.7. It's said better at [334:31-34].

397:2,5,8 ["A ... has the scope of a derived-type definition." Which derived-type definition? Editor: "of a" => "of its" thrice.]

397:5 [Editor: Delete the first "derived".]

397:23 [I don't think that scoping units make references.
Editor: "making" => "in which the"; insert "occurs" after the first "procedure".]

397:25+ Add the following paragraph:
"A variable that appears as a DO variable of an implied-DO in a DATA statement or an array constructor, as a dummy argument in a statement function statement, or as an *index-name* in a FORALL statement is a statement entity. A variable that appears as an *index-name* in a FORALL construct or an *associate-name* in a SELECT TYPE or ASSOCIATE construct is a construct entity."

397:33 [Editor: "index-name" => "*index-name*".]

398:11-12 [Implies that a FORALL construct has only one *index-name*. Editor: "the *index-name*" => "any of its *index-name*s"; "a nested ... *index-name*" => "an *index-name* of a FORALL statement or FORALL construct shall not be the same as an *index-name* of a containing FORALL construct".]

Do NOT do this edit. Reason: Not needed.
399:12 [It isn't clear that entities *not* named in IMPORT statements are not accessible by host association. Editor: Insert ", and only to those entities" after "body".]

Do NOT make this edit. Reason: The procedure name declared by the interface body is a global name.
399:18 [An interface body is a local entity. Editor: ", procedure-declaration-stmt, or interface-body" => "or procedure-declaration-stmt".]

Do NOT make this edit. Reasons: We like the list as it is. An interface body does not have a name and this passage deals with named objects.

399:20-400:2 [Local entities are already listed at [394:3-10]. We don't need to list them again. Editor: "A name ... nongeneric name" => "Any entity of the host that has the same nongeneric name as a local entity". If we don't remove the list, at least make it consistent with [394:3-10] by adding "an interface-body" somewhere. Then, put the list in the same order as at [394:3-10] so it's easier to verify we're not committing the error Dick Weaver observed: "Say it twice, say it wrong at least once."]

Do NOT make this edit. Reason: We like the note as it is.

400:8+1-3 [The essence of Note 16.10 is on the previous page, and, with the edit for [399:12], almost verbatim. Editor: Delete Note 16.10.]

Do NOT make this edit. Reason: The existing text makes it clear when an EXTERNAL statement or a TYPE statement with the EXTERNAL attribute is required in the host scoping unit.

400:9 [If an external or dummy procedure has an explicit interface, it thereby has the EXTERNAL attribute, so "with an implicit interface" is redundant. Editor: Delete it.]

400:13,16 [Which scoping unit is "that" scoping unit? It could be the inner one, a host scoping unit, or a module. Editor: "that scoping unit" => "the scoping unit from which it is accessed" twice.]

2 Is this a spec change, a stealth interp, or just clean-up?

The following edits, if implemented, would specify that the appearance of a name as the dummy argument of a statement function does not constitute an implicit declaration of a variable in the scoping unit that contains it. Is this a spec change, a stealth interp, or just clean-up?

Do NOT do any of the following edits. JoR believes that the existing text in 02-007r2 is fine just the way it is and no changes are needed.

397:30-32 [Now that we have defined "statement entity" and "construct entity" we can simplify the part about it not being an implicit declaration of a variable in the scoping unit that contains its statement or construct (and also specify it for statement function dummy arguments at the same time). Editor: Delete.]

397:36-38 [Editor: Delete "The appearance ... construct."]

397:41+ Same ¶ The appearance of a name as the name of a statement or construct entity is not an implicit declaration of a variable of that name whose scope is the scoping unit that contains its statement or construct. A statement or construct entity is not accessible outside of its statement or construct.

3 Not sure what to do

Do nothing with any of the issues in this section.

[394:4] refers to "Named variables that are not statement or construct entities." I can't find where statement and construct entities are defined to be variables. So "named variables that are not statement or construct entities" appears to be just "named variables". That's probably wrong, so we should perhaps say somewhere that statement and construct entities are variables.

[394:19-395:6] doesn't make sense. [394:19-20] claims to be about the names of local entities. Then the next three items introduced by that sentence are about procedure names within their subprograms. A name for an external procedure isn't a local name within its subprogram, but the parenthetic remark says that one usage of the procedure name is allowed only for module or internal procedures, implying that the other case, an external procedure, is a local entity.

If a procedure is recursively referenced from within itself, does the procedure name *identify* that reference?

The parts about common blocks appear to be adequately covered by 16.2.1.

The "except" part of the intro to these items doesn't make sense: The appearance of a procedure name within that procedure isn't "in another scoping unit."

If items (1) – (3) actually do make sense, we need to say something about referring to the procedure name to get the interface for a procedure pointer or deferred type-bound procedure declared within the subprogram. These items appear to prohibit such usages.

References

02-007r2, Fortran 2000 Draft

[End of J3 / 02-221r1]