Interpretations 68 & 70

To: J3
From: Interpretations Subgroup
Date: May 30, 2000
Subject: Interpretations 68 & 70

The Interpretations Subgroup reviewed the ballot results of Interpretations 68 & 70.

Interpretations decided that no substantive changes needed to be made to the edits for Interpretation 68.

In addition to the edits passed at Meeting 152, Interpretations added the following two edits to Interpretation 70. The original edits missed two instances of the phrase "specification expression" at [54:36] and [56:34].

NUMBER: 000068
TITLE: Asterisks as I/O units
KEYWORDS: Asterisk, I/O, unit
DEFECT TYPE: Erratum
STATUS: Passed by J3 letter ballot

Question:

1. Does the Fortran 95 standard require the external unit corresponding to the I/O unit identified by an asterisk for input or output to be the same as the external unit identified by some fixed nonnegative integer value?

2. Can the I/O unit identified by an asterisk for input or output correspond to more than one external unit?

3. If the external unit identified by an integer value that corresponds to the I/O unit identified by an asterisk for input or output is closed, does that cause the I/O unit identified by an asterisk to become disconnected?

Answer:

1. No.

2. No.

3. Yes.

Discussion:

The submitter states: At least one Fortran 95 implementation uses -1 as the value of the I/O unit identified by an asterisk. A carefully constructed INQUIRE statement can expose this value to the user. Many users expect I/O to the units identified by asterisks to continue to work even after the corresponding units identified by integer values have been closed.

1. There is no requirement in the standard that the asterisk correspond to an external-file-unit.

2. For the units identified by the asterisk, the text of section 9.3.2 does not allow two or more units to be connected simultaneously to the same external device or file [139:8-9]. An edit is supplied to clarify this situation.

3. There might not be an external-file-unit that corresponds to the io-unit specified by an asterisk, as clarified by the edit below. If there is, it is permissible to execute a CLOSE statement on them as on any other unit.

REFERENCES: ISO/IEC 1539-1:1997(E), Sections 9.3 and 9.4.4.2

EDITS:
For Fortran 95:
[138:34+] Add the following text to the end of the last paragraph before section 9.3.1:
"An asterisk used in an input statement may identify the same *io-unit* as a particular *external-file-unit*. An asterisk used in an output statement may identify the same *io-unit* as another particular *external-file-unit*."

There are no edits required for Fortran 2000.

SUBMITTED BY: Robert Corbett

HISTORY: 99-192 m150 Submitted
99-215r1 m150 approved uc
00-208 m153 Passed by J3 letter ballot

NUMBER: 000070
TITLE: Asymmetry between constant specification and initialization expressions
KEYWORDS: Initialization expressions; specification expressions
DEFECT TYPE: Erratum
STATUS: Passed by J3 letter ballot

QUESTION:
Consider the following programs.

```
   PROGRAM P1
     REAL :: B = 4.0*ATAN(1.0)
     PRINT *, B
   END PROGRAM P1

   PROGRAM P2
     INTEGER :: A(INT(4*ATAN(1.0))) = 17
     PRINT *, A
   END PROGRAM P2
```

According to 7.1.6.1 program unit P1 is not standard-conforming because of the reference to the intrinsic function ATAN which is not permitted in an initialization expression.

According to 7.1.6.2 program unit P2 is standard-conforming; the reference to the intrinsic function ATAN is allowed by item (8) in the definition of a restricted expression. Expressions in the array bounds of an initialized entity are only required to be constant specification expressions, not initialization expressions.

Was it the committee's intent to permit ATAN to appear in the array bounds of an initialized entity but not in the initialization value?

ANSWER:

No, this was not the intent. These expressions should have been described as initialization expressions instead of as constant expressions.

This error also occurs for the definitions of an automatic entity, common block definitions and component definitions.

The edits below change all of these to require initialization expressions instead of constant expressions.

EDIT:

[39:15] Change "a constant specification" to "an initialization".
{Fix array components.}

[39:23] Change "a constant specification" to "an initialization".
{Fix character string components.}

[40:30] Change "a constant" to "an initialization".
{Fix note.}

[48:47-48] Change "may be a nonconstant expression provided the specification expression"
to "shall be an initialization expression unless it".
[49:1-3] Delete "If a ... nonconstant expression."
[49:4-5] Change "such a nonconstant expression"
to "a <specification-expr> that is not an initialization expression".

Craig T. Dedo

{Fix definition of "automatic object".}

[49:9] Change "a nonconstant expression"
to "an expression that is not an initialization expression".
{Fix evaluation time for character length.}

[51:33] Change "a constant specification" to "an initialization".
{Fix statement function character lengths.}

[54:33] Change "nonconstant specification" to "not initialization".
{Fix automatic array definition.}

[54:34] Change "nonconstant specification" to "not initialization".
{Fix evaluation time for explicit-shape array bounds.}

[54:36] Change "the specification" to "any bounds".

[56:32] Change "nonconstant specification" to "not initialization".
{Fix evaluation time for assumed-size array bounds.}

[56:34] Change "the specification" to "any bounds".

[69:3-4] Change "a constant specification expression (7.1.6.2)"
to "an initialization expression (7.1.6.1)".
{Fix common block array-specs.}

[192:26] Change "a constant" to "an initialization".
{Fix characteristics of function results.}

SUBMITTED BY: Henry Zongaro
HISTORY: 99-178 m150 submitted
99-216r1 m150 approved uc
00-133 m152 additional edit, approved uc
00-208 m153 Passed by J3 letter ballot

[End of J3 / 00-208]