Information technology – Programming languages – Fortran

TECHNICAL CORRIGENDUM 3

Technical corrigendum 3 to international Standard ISO/IEC 1539:1991 (E) was prepared by Joint Technical Committee ISO/IEC JTC1, Information technology.

Page 19

Subclause 3.2.1

Change “Keywords” to “Statement keywords”, twice.

Pages 22 and 23

Subclause 3.3.1.2

Change the first sentence to:

> The character ″;″ terminates a statement, except when the ″;″ appears in a character context or in a comment. This optional termination allows another statement to begin following the ″;″ on the same line. A ″;″ must not appear as the first nonblank character on a line.

and delete the last sentence of this subclause.

Page 23

Subclause 3.3.2.2

Change the first sentence to:

> The character ″;″ terminates a statement, except when the ″;″ appears in a character context, in a

> comment, or in character position 6. This optional termination allows another statement to begin following the ″;″ on the same line. A ″;″ must not appear as the first nonblank character on a line, except in character position 6.

and delete the last sentence of this subclause.

Page 38

Clause 4.5

In the second constraint following R435, change “the same type and type parameters.” to: “the same type and kind type parameter.”

Add the following paragraph after the constraints:

> If the *ac-value* expressions are of type character, each *ac-value* expression in the *array-constructor* must have the same length type parameter.

Page 44

Subclause 5.1.2.3

Following R511, add the following constraint after the existing constraint:

Constraint: A dummy argument with the INTENT(IN) attribute, or subobject of such a dummy argument, must not appear

(1) As the *variable* of an *assignment-stmt*,

(2) As a DO variable or implied-DO variable,

(3) As an *input-item* in a *read-stmt*,

(4) As a *variable-name* in a *namelist-stmt* if the *name-list-group-name* appears in a NML= specifier in a *read-stmt*,

(5) As an *internal-file-unit* in a *write-stmt*,

(6) As an IOSTAT= or a SIZE= specifier in an input/output statement,

(7) As an inquiry specifier variable in an INQUIRE statement,

(8) As a *stat-variable* in an *allocate-stmt* or *deallocate-stmt*,

(9) As an actual argument in a reference to a procedure with an explicit interface when the associated dummy argument has the INTENT(OUT) or INTENT(INOUT) attribute, or

(10) In an *assign-stmt*.

Page 48

Subclause 5.1.2.8

Delete the second sentence, “An object ... with it.”.

Subclause 5.1.2.10

In the first line, replace “an object name” with “a function name”.

Subclause 5.1.2.11

In the first line, replace “an object name” with “a function name”.

Page 57

Subclause 5.5.1

On line 2 of page 57, change “derived type or” to “derived type, an object”.

Add a new constraint after the existing constraints, including the one added by corrigendum 2:

> Constraint: A *substring* must not have length zero.

Page 59

Subclause 5.5.2.3

Insert as the (new) last paragraph of subclause 5.5.2.3: “An object with the TARGET attribute may become storage associated only with another object that has the TARGET attribute and the same type and type parameters.”.

Pages 77 and 78

Subclause 7.1.6.1

In item (1) of the first list, change “constant where each subscript, section subscript, substring starting point, and substring ending point is a constant expression.” to: “constant,”.

In item (2) of the first list, change “either constant expressions or implied-DO variables,” to “constant expressions,”.

In item (6) of the first list (which was altered by corrigendum 2), delete the last “or”.

In the first list, replace item (7) by:

(7) An implied-DO variable within an array constructor where the bounds and strides of the corresponding implied-DO are constant expressions, or

(8) A constant expression enclosed in parentheses,

and where each subscript, section subscript, substring starting point, and substring ending point is a constant expression.

In item (1) of the second list, change “constant where each subscript, section subscript, substring starting point, and substring ending point is an initialization expression,” to “constant,”

In item (2) of the second list, change: “either initialization expressions or implied-DO variables,” to: “initialization expressions,”.

In item (6) of the second list (which was altered by corrigendum 2), delete the last “or”.

In the second list, replace item (7) by:

(7) An implied-DO variable within an array constructor where the bounds and strides of the corresponding implied-DO are initialization expressions, or

(8) An initialization expression enclosed in parentheses,

and where each subscript, section subscript, substring starting point, and substring ending point is an initialization expression.

Page 79

Subclause 7.1.6.2

In item (5) of the list, change: “either restricted expressions or implied-DO variables,” to: “restricted expressions,”.

In item (9) of the list (which was altered by corrigendum 2), delete the last “or”.

Add as item 10 of the list:

(10) An implied-DO variable within an array constructor where the bounds and strides of the corresponding implied-DO are restricted expressions, or

and renumber the rest.

Add to the end of the paragraph ahead of R734, "A **constant specification expression** is a specification expression that is also a constant expression.".

Pages 115 and 116

Subclause 9.3.4

In line 3 of the sixth paragraph, change "currently in effect." to:

> currently in effect. If the POSITION= specifier is present in such an OPEN statement, the value specified must not disagree with the current position of the file. If the STATUS= specifier is included in such an OPEN statement, it must be specified with a value of OLD.

After the sixth paragraph, add the paragraph:

> Note that a STATUS= specifier with a value of OLD is always allowed when the file to be connected to the unit is the same as the file to which the unit is connected. In this case, if the status of the file was SCRATCH before execution of the OPEN statement, the file will still be deleted when the unit is closed, and the file is still considered to have the status SCRATCH.

On line 3 of page 116, delete "OLD," and add the sentence "If the STATUS= specifier has the value OLD, the FILE= specifier must be present unless the unit is currently connected and the file connected to the unit exists.".

Page 117

Subclause 9.3.4.8

In line 2, change the first "WRITE" to "WRITE, PRINT,".

Subclause 9.3.4.9

In lines 5 and 6, delete the sentence "If APOSTROPHE or QUOTE is specified, a *kind-param* and underscore will be used to precede the leading delimiter of a nondefault character constant.".

Page 153

Subclause 10.9.1.3

In the second line of the third paragraph, change "blanks," to "blanks, equals,".

Page 158

Subclause 11.3.2

In the first sentence after the constraints change "the local name is the *use-name*" to "the local name of a named entity is the *use-name*".

In the second sentence of the fourth paragraph after the constraints change "and the *rename-list*s and *only-list*s are interpreted as a single concatenated *rename-list*" to ", the *rename-list*s and renames in *only-list*s are interpreted as a single concatenated *rename-list*, and entities in the remaining *only-list* items are accessible by those *use-name*s or *access-id*s (they may also be accessible by one or more renames)".

Near the bottom of the page, ahead of "Examples:" add two new paragraphs:

> A procedure with an implicit interface and public accessibility must explicitly be given the EXTERNAL attribute in the scoping unit of the module; if it is a function, its type and type parameters must be explicitly declared in a type declaration statement in that scoping unit.
>
> An intrinsic procedure with public accessibility must explicitly be given the INTRINSIC attribute in the scoping unit of the module or be used as an intrinsic procedure in that scoping unit.

Pages 163 and 164

Subclause 12.1.2.2.1

In the last line of page 163, change "A name that appears in the scoping unit as an *external-name* in an *external-stmt*" to "A name that is declared to be an external procedure name (by an *external-stmt* or an *interface-body*), or that appears as a *module-name* in a *use-stmt*".

In item (2) of the list on page 164, change "A *function-name* in a *function-stmt*, in a *stmt-function-stmt*, or" to "A *function-name* in a *stmt-function-stmt* or".

In the list on page 164, delete items (1), (3) and (4) and renumber the rest.

Following the list on page 164, ahead of the sentence "Entities that are local (14.1.2) to a procedure are not accessible to its host." add

> If a scoping unit contains a subprogram or a derived type definition, the name of the subprogram or derived type is the name of a local entity. Any entity of the host of this scoping unit that has a nongeneric name that is the same as the name of the subprogram or derived type is inaccessible.

Page 171

Subclause 12.3.2.3

Add to the end of the first paragraph after the constraint "In a scoping unit, a name can appear as both the name of a generic intrinsic procedure in an INTRINSIC statement and as the name of a generic interface if procedures in the interface and the specific intrinsic procedures are all functions or all subroutines (14.1.2.3).".

Page 173

Subclause 12.4.1.1

At the end of the second paragraph of page 173 delete the sentence, which was added by corrigendum 2, "If the dummy argument has the TARGET attribute and the actual argument has the TARGET attribute but is not an array section with a vector subscript, the dummy and actual arguments must have the same shape."

The fourth paragraph of page 173, commencing "If the actual", was replaced by corrigendum 2. This replacement text must be discarded and replaced by:

> If the dummy argument does not have the TARGET or POINTER attribute, any pointers associated with the actual argument do not become associated with the corresponding dummy argument on invocation of the procedure. If such a dummy argument is associated with a dummy argument with the TARGET attribute, whether any pointers associated with the original actual argument become associated with the dummy argument with the TARGET attribute is processor dependent.
>
> If the dummy argument has the TARGET attribute and is either a scalar or an assumed-shape array, and the corresponding actual argument has the TARGET attribute but is not an array section with a vector subscript:
>
> (1) Any pointers associated with the actual argument become associated with the corresponding dummy argument on invocation of the procedure.
>
> (2) When execution of the procedure completes, any pointers associated with the dummy argument remain associated with the actual argument.
>
> If the dummy argument has the TARGET attribute and is an explicit-shape array or is an assumed-size array, and the corresponding actual argument has the TARGET attribute but is not an array section with a vector subscript:
>
> (1) On invocation of the procedure, whether any pointers associated with the actual argument become associated with the corresponding dummy argument is processor dependent.

> (2) When execution of the procedure completes, the pointer association status of any pointer that is pointer associated with the dummy argument is processor dependent.

> If the dummy argument has the TARGET attribute and the corresponding actual argument does not have the TARGET attribute or is an array section with a vector subscript, any pointers associated with the dummy argument become undefined when execution of the procedure completes.

Subclause 12.4.1.2

Replace the second paragraph with

> If the interface of the dummy procedure is explicit, the characteristics listed in 12.2 must be the same for the associated actual procedure as for the corresponding dummy procedure.

Page 179

Subclause 12.5.2.8

Add the following to the numbered list after item (5), which was added by corrigendum 1:

> (6) If it is a pointer, it must not be supplied as an actual argument corresponding to a nonpointer dummy argument other than as the argument of the PRESENT intrinsic function.

In the line following the numbered list, which was altered by corrigendum 1, change “in (5)” to “in the list”.

Pages 180 and 181

Subclause 12.5.2.9

Replace the first two lines of item (1) by

> No action that affects the allocation status of the entity may be taken. Action that affects the value of the entity or any part of it must be taken through the dummy argument unless
>
> (a) the dummy argument has the POINTER attribute,
>
> (b) the part is all or part of a pointer subobject, or
>
> (c) the dummy argument has the TARGET attribute, the dummy argument does not have INTENT(IN), the dummy argument is a scalar object or an assumed-shape array and the actual argument is a target other than an array section with a vector subscript.
>
> For example, in

In the line following the line “`DEALLOCATE (A)`” change “availability of A” to “allocation of B”.

Replace the two lines following the line “`DEALLOCATE(B)`” by

> also would not be permitted. If B were declared with the POINTER attribute, either of the statements
>
> `DEALLOCATE(A)`
>
> and
>
> `DEALLOCATE(B)`
>
> would be permitted, but not both.

In the second line of the paragraph on page 180 that commences “Note that”, after “the same procedure” add “and the dummy arguments have neither the POINTER nor the TARGET attribute”.

In line 8 of page 181, after “`CALL SUB(B,C)`”, add

```
! The dummy arguments of SUB are neither pointers nor targets
```

On page 181, replace the first three lines of item (2) by

> If the value of any part of the entity is affected through the dummy argument, then at any time

during the execution of the procedure, either before or after the definition, it may be referenced only through that dummy argument unless

(a) the dummy argument has the POINTER attribute,

(b) the part is all or part of a pointer subobject, or

(c) the dummy argument has the TARGET attribute, the dummy argument does not have INTENT(IN), the dummy argument is a scalar object or an assumed-shape array and the actual argument is a target other than an array section with a vector subscript.

For example, in

Page 182

Subclause 12.5.4

In the first sentence of the first constraint, change “The *scalar-expr* may be composed only of” to “The primaries of the *scalar-expr* must be”.

In the second sentence of the first constraint, which was supplied in corrigendum 1, change “or be a transformational intrinsic,” to “unless it is an intrinsic, the function must not be a transformational intrinsic,”

Page 183

Subclause 13.2.2

Add the following sentences to the end of the paragraph:

In a reference to the intrinsic subroutine MVBITS, the actual arguments corresponding to the TO and FROM dummy arguments may be the same variable. Apart from this, the actual arguments in a reference to an intrinsic subroutine must satisfy the restrictions of 12.5.2.9.

Clause 13.3

Change the title to be

13.3 Arguments to intrinsic procedures

Add the following as a new paragraph at the end of clause 13.3:

The dummy arguments of the specific intrinsic procedures in 13.12 have INTENT(IN). The nonpointer dummy arguments of the generic intrinsic procedures in 13.13 have INTENT(IN) if the intent is not stated explicitly.

Pages 198 and 199

Subclause 13.13.13

In the description of the TARGET dummy argument, following “must be a pointer or target”, add “, and have the same type, type parameters, and rank as POINTER”.

Replace the text for *Cases (ii)* and *(iii)* by:

Case (ii): If TARGET is present and is a scalar target, the result is true if TARGET is not a zero-sized storage sequence and the target associated with POINTER occupies the same storage units as TARGET. Otherwise, the result is false. If POINTER is disassociated, the result is false.

Case (iii): If TARGET is present and is an array target, the result is true if the target associated with POINTER and TARGET have the same shape, are neither of size zero nor arrays whose elements are zero-sized storage sequences, and occupy the same storage units in array element order. Otherwise, the result is false. If POINTER is disassociated, the result is false.

Case (iv): If TARGET is present and is a scalar pointer, the result is true if the target associated with POINTER and the target associated with TARGET are not zero-sized storage sequences and they occupy the same storage units. Otherwise, the result is false. If either POINTER or TARGET is disassociated, the result is false.

Case (v): If TARGET is present and is an array pointer, the result is true if the target associated with POINTER and the target associated with TARGET have the same shape, are neither of size zero nor arrays whose elements are zero-sized storage sequences, and occupy the same storage units in array element order. Otherwise, the result is false. If either POINTER or TARGET is disassociated, the result is false.

Page 205

Subclause 13.13.27

Replace the text of the “Result Value” section by:

Result Value. The result has the value REAL (A, KIND (0.0D0)).

Page 228

Subclause 13.13.84

In the text for PUT, replace the sentence “It is used by the processor ...” by “It is used in a processor-dependent manner to compute the seed value accessed by the pseudorandom number generator.”

In the text for GET, replace the sentence “It is set by the processor ...” by “It is assigned the current value of the seed.”

Add the following text before the examples:

The pseudorandom number generator accessed by RANDOM_SEED and RANDOM_NUMBER maintains a seed that is updated during the execution of RANDOM_NUMBER and that may be specified or returned by RANDOM_SEED. Computation of the seed from argument PUT is performed in a processor dependent manner. The value returned by GET need not be the same as the value specified by PUT in an immediately preceding reference to RANDOM_SEED. For example, following execution of the statements

```
CALL RANDOM_SEED(PUT=SEED1)
CALL RANDOM_SEED(GET=SEED2)
```

SEED2 need not equal SEED1. When the values differ, the use of either value as the PUT argument in a subsequent call to RANDOM_SEED must result in the same sequence of pseudorandom numbers being generated. For example, after execution of the statements

```
CALL RANDOM_SEED(PUT=SEED1)
CALL RANDOM_SEED(GET=SEED2)
CALL RANDOM_NUMBER(X1)
CALL RANDOM_SEED(PUT=SEED2)
CALL RANDOM_NUMBER(X2)
```

X2 equals X1.

Pages 232

Subclause 13.13.93

In lines 4-5 of the paragraph prefaced with **Result Value**, change “–1 if ... neither is available.” to “–1 if the processor does not support a real data type with a precision greater than or equal to P, –2 if the processor does not support a real data type with an exponent range greater than or equal to R, and –3 if neither is supported.”.

Pages 241 and 242

Subclause 14.1.2

In line 1 of the second paragraph, after "(14.1.2.1)", add ", an external procedure name that is also a generic name (12.3.2.1),".

On page 242, at the end of the last sentence of subclause 14.1.2, replace "." by:

> , except in the following cases:
>
> (1) The name that appears as a *subroutine-name* in a *subroutine-stmt* has limited use within the scope established by the *subroutine-stmt*. It can be used to identify recursive references of the subroutine or to identify the name of a common block (the latter is possible only for internal and module subroutines).
>
> (2) The name that appears as a *function-name* in a *function-stmt* has limited use within the scope established by that *function-stmt*. It can be used to identify the result variable, to identify recursive references of the function, or to identify the name of a common block (this last use is possible only for internal and module functions).
>
> (3) The name that appears as an *entry-name* in an *entry-stmt* has limited use within the scope of the subprogram in which the *entry-stmt* appears. It can be used to identify the name of a common block (if the ENTRY statement is in a module subprogram), to identify recursive references, or if the subprogram is a function to identify the result variable.

Pages 242 and 243

Subclause 14.1.2.3

In the third line, change "When an intrinsic procedure, operator, or" to "When an intrinsic operator or".

In line 2 of the third paragraph, change "and at least one of them must have a nonoptional dummy argument that" to

> and
>
> (1) one of them must have more nonoptional dummy arguments of a particular data type, kind type parameter, and rank than the other has dummy arguments (including optional dummy arguments) of that data type, kind type parameter, and rank; or
>
> (2) at least one of them must have a nonoptional dummy argument that

and indent numbers (1) and (2) to be a sublist of list item (2), changing them to be (a) and (b) respectively.

In the text after the example on page 243, change "(1)" to "(2)(a)" twice and change "(2)" to "(2)(b)" twice.

Add as a new paragraph at the end of the subclause:

> If a generic name is the same as the name of a generic intrinsic procedure, the generic intrinsic procedure is not accessible if the procedures in the interface and the intrinsic procedure are not all functions or not all subroutines. If a generic invocation is consistent with both a specific procedure from an interface and an accessible generic intrinsic procedure, it is the specific procedure from the interface that is referenced.

Page 245

Clause 14.4

In line 1, after "A defined operator", add "that is not an extended intrinsic operator".

Clause 14.5

In line 2, after "operations", add "or replace the intrinsic derived type assignment operation".

Page 269

Subclause C.5.3

In line 2, change “solely” to “primarily”.

In lines 4 to 7, delete the sentence “The rule ...attribute.”.

Page 291

Subclause C.12.7

Replace the sentence “This ... storage.” on lines 3 to 5 by “The restrictions on entities associated with dummy arguments are intended to facilitate a variety of optimizations in the translation of the procedure, including implementations of argument association in which the value of an actual argument that is neither a pointer nor a target is maintained in a register or in local storage.”

Page 292

Subclause C.12.8

Replace the paragraph “When execution of a procedure completes, ... subscript.”, which was added by corrigendum 2, by:

> When execution of a procedure completes, any pointer that remains defined and that is associated with a dummy argument that has the TARGET attribute and is either a scalar or an assumed-shape array, remains associated with the corresponding actual argument if the actual argument has the TARGET attribute and is not an array section with a vector subscript.