



**INTERNATIONAL STANDARD ISO/IEC 1539-1:2010**  
**TECHNICAL CORRIGENDUM 2**

Published 2013-06-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION  
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

# **Information technology — Programming languages — Fortran**

## **Part 1: Base language**

### **TECHNICAL CORRIGENDUM 2**

*Technologies de l'information — Langages de programmation — Fortran*

*Partie 1: Langage de base*

*RECTIFICATIF TECHNIQUE 2*

Technical Corrigendum 2 to ISO/IEC 1539-1:2010 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

**ISO/IEC 1539-1:2010 - TECHNICAL CORRIGENDUM 2****Introduction**

Following the second sentence in the fifth item in the bulleted list (Data usage and computation), insert: "Multiple allocations are permitted in a single ALLOCATE statement with SOURCE=."

**Subclause 1.6.2**

Replace the two paragraphs added to the subclause in Technical Corrigendum 1 by the following six paragraphs:

Fortran 2003 permitted a sequence type to have type parameters; that is not permitted by this part of ISO/IEC 1539.

Fortran 2003 specified that array constructors and structure constructors of finalizable type are finalized. This part of ISO/IEC 1539 specifies that these constructors are not finalized.

The form produced by the G edit descriptor for some values and some I/O rounding modes differs from that specified by Fortran 2003.

Fortran 2003 required an explicit interface only for a procedure that was actually referenced in the scope, not merely passed as an actual argument. This part of ISO/IEC 1539 requires an explicit interface for a procedure under the conditions listed in 12.4.2.2, regardless of whether the procedure is referenced in the scope.

Fortran 2003 permitted the result variable of a pure function to be a polymorphic allocatable variable, or to be finalizable by an impure final subroutine. These are not permitted by this part of ISO/IEC 1539.

Fortran 2003 permitted an INTENT(OUT) argument of a pure subroutine to be polymorphic; that is not permitted by this part of ISO/IEC 1539.

**Subclause 1.6.3**

Add the following item at the end of the bulleted list:

- The form produced by the G edit descriptor with  $d$  equal to zero differs from that specified by Fortran 95 for some values.

**Subclause 1.6.4**

In the fourth paragraph of the subclause, replace the full stop at the end of the third bulleted item by a semicolon and add a fourth item:

- the G edit descriptor with  $d$  equal to zero for some values.

**Subclause 4.4.2.3**

In the third paragraph of the subclause, in Note 4.8 change "can distinguish" to "distinguishes".

**Subclause 4.5.2.3**

Replace constraint C436 by:

- C436 (R425) If SEQUENCE appears, each data component shall be declared to be of an intrinsic type or of a sequence type, the derived type shall not have type parameters, and a *type-bound-procedure-part* shall not appear.

**Subclause 4.5.2.4**

In the second sentence of the second paragraph of the subclause, delete “type parameters and”.

**Subclause 4.5.6.1**

In the second sentence of constraint C480, insert “noncoarray,” before “nonpointer”.

**Subclause 4.5.6.3**

To the second paragraph of the subclause (which was paragraph 1 prior to the edits of Technical Corrigendum 1), append the new sentence:

If an error condition occurs during deallocation, it is processor dependent whether finalization occurs.

Replace the seventh paragraph of the subclause (which was paragraph 8 prior to the edits of Technical Corrigendum 1) by the following new paragraph. This supersedes the substitute paragraph given in Technical Corrigendum 1.

When a procedure is invoked, a nonpointer, nonallocatable INTENT(OUT) dummy argument of that procedure is finalized before it becomes undefined. The finalization caused by INTENT(OUT) is considered to occur within the invoked procedure; so for elemental procedures, an INTENT(OUT) argument will be finalized only if a scalar or elemental final subroutine is available, regardless of the rank of the actual argument.

**Subclause 4.8**

In constraint C4105 in the first paragraph of the subclause, change “all *ac-value* expressions in the *array-constructor* shall be of that derived type and” to “the declared type of each *ac-value* expression in the *array-constructor* shall be that derived type and”.

After constraint C4106, insert the following new constraint:

C4106a (R472) The declared type of an *ac-value* shall not be abstract.

In the second paragraph of the subclause, change “each *ac-value* expression in the array constructor shall have the same length type parameters;” to “corresponding length type parameters of the declared type of each *ac-value* expression shall have the same value;”.

In the third paragraph of the subclause, after “Each value is converted to the” insert “type and”.

**Subclause 5.3.7**

In the first paragraph of the subclause, change “can only be argument associated with a contiguous effective argument” to “is contiguous”.

**Subclause 5.3.10**

In constraint C541 change “An entity” to “A dummy argument of a nonintrinsic procedure”.

**Subclause 5.4.7**

In the fourth paragraph of the subclause, replace constraint C566 by:

C566 (R536) A *data-stmt-object* that is a *variable* shall be a *designator*. Each subscript, section subscript, substring starting point, and substring ending point in the *variable* shall be a constant expression.

**Subclause 5.5**

In the final sentence of the third paragraph of the subclause, change “an internal or module procedure” to “a BLOCK construct, internal subprogram, or module subprogram”.