INTERNATIONAL STANDARD

ISO/IEC 1539-1

Third edition
2010-10-15

Information technology — Programming languages — Fortran —

Part 1: Base language

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

Partie 1: Langage de base

Reference number
ISO/IEC 1539-1:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

List of Tables

Foreword

1 ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and nongovernmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

2 International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

3 The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

4 Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

6 This third edition cancels and replaces the second edition (ISO/IEC 1539-1:2004), which has been technically revised. It also incorporates the Technical Corrigenda ISO/IEC 1539-1:2004/Cor. 1:2006, ISO/IEC 1539-1:2004/Cor. 2:2007, ISO/IEC 1539-1:2004/Cor. 3:2008, and ISO/IEC 1539-1:2004/Cor. 4:2009, and the Technical Report ISO/IEC TR 19767:2005.

7 ISO/IEC 1539 consists of the following parts, under the general title *Information technology — Programming languages — Fortran*:

8 — *Part 1: Base language*

9 — *Part 2: Varying length character strings*

10 — *Part 3: Conditional compilation*

Introduction

1 This part of ISO/IEC 1539 comprises the specification of the base Fortran language, informally known as Fortran 2008. With the limitations noted in 1.6.2, the syntax and semantics of Fortran 2003 are contained entirely within Fortran 2008. Therefore, any standard-conforming Fortran 2003 program not affected by such limitations is a standard-conforming Fortran 2008 program. New features of Fortran 2008 can be compatibly incorporated into such Fortran 2003 programs, with any exceptions indicated in the text of this part of ISO/IEC 1539.

2 Fortran 2008 contains several extensions to Fortran 2003; some of these are listed below.

- Module enhancements:
  Submodules provide additional structuring facilities for modules. Data objects and procedure pointers declared in a module implicitly have the SAVE attribute.
- Parallel execution:
  Coarrays and synchronization constructs support parallel programming using a single program multiple data (SPMD) model.
- Performance enhancements:
  The DO CONCURRENT construct provides a means for the program to specify that individual loop iterations have no interdependencies. The CONTIGUOUS attribute provides a means for the program to specify restrictions on the storage layout of pointer targets and assumed-shape dummy arguments.
- Data declaration:
  The maximum rank has been increased to 15. A processor is required to support at least one kind of integer with a range of at least 18 decimal digits. An allocatable component can be of recursive type. A named constant array's shape can be implied by its value. A pointer can be initially associated with a target. Subscripts and nested implied-do limits inside a *data-implied-do* can be any constant expression instead of being limited to combinations of constants, implied-do variables, and intrinsic operations. A FORALL index variable can have its type and kind explicitly declared within the construct. The TYPE keyword can be used to declare entities of intrinsic type. Multiple type-bound procedures can be declared in a single type-bound procedure statement.
- Data usage and computation:
  A structure constructor can omit the value for an allocatable component. SOURCE= in an ALLOCATE statement can give an array variable the bounds as well as the value of an expression. MOLD= in an ALLOCATE statement can give a polymorphic variable the shape, type, and type parameters of an expression without copying the value. The real and imaginary parts of a complex entity can be accessed independently with a component-like syntax. Intrinsic assignment to an allocatable polymorphic variable is allowed. A pointer function reference can denote a variable in any variable definition context. Some restrictions on the use of dummy arguments in elemental subprograms have been removed.
- Input/output:
  NEWUNIT= in an OPEN statement automatically selects a unit number that does not interfere with other unit numbers selected by the program. The G0 edit descriptor and unlimited format control ease writing output in comma-separated-value (CSV) format. Recursive data transfers are allowed on distinct units.
- Execution control:
  The BLOCK construct can contain declarations of objects with construct scope. The EXIT statement can transfer control from within more named executable constructs. The STOP statement has been changed to accept a constant expression instead of merely a literal constant, and to encourage the processor to provide the integer stop code (if it appears) as a termination status (where that makes sense). The ERROR STOP statement initiates error termination.
- Intrinsic procedures:
  - The intrinsic functions ACOS, ASIN, ATAN, COSH, SINH, TAN, and TANH can have arguments of type complex.
  - The new intrinsic functions ACOSH, ASINH, and ATANH calculate the inverse hyperbolic cosine, sine, and tangent respectively.
  - The intrinsic function ATAN2 can be referenced by the name ATAN.
  - The new intrinsic subroutines ATOMIC_DEFINE and ATOMIC_REF define and reference a variable

atomically.
- The new intrinsic functions BESSEL_J0, BESSEL_J1, BESSEL_JN, BESSEL_Y0, BESSEL_Y1, and BESSEL_YN calculate Bessel functions.
- The new intrinsic functions BGE, BGT, BLE, and BLT perform bitwise comparisons.
- The new intrinsic functions DSHIFTL and DSHIFTR calculate combined left and right shifts.
- The new intrinsic functions ERF, ERFC, and ERFC_SCALED calculate the error function and its complement.
- The new intrinsic subroutine EXECUTE_COMMAND_LINE allows a program to start another program.
- The new intrinsic function FINDLOC searches an array for a value.
- The intrinsic functions LGE, LGT, LLE, and LLT can have arguments of ASCII kind.
- The new intrinsic functions GAMMA and LOG_GAMMA calculate the gamma function and its log.
- The new intrinsic function HYPOT calculates the Euclidean distance.
- The new intrinsic functions IALL, IANY, and IPARITY reduce an array with the bitwise AND, bitwise OR, and bitwise exclusive OR functions respectively.
- The new intrinsic function IMAGE_INDEX converts cosubscripts to an image index.
- The new intrinsic functions LCOBOUND and UCOBOUND return the cobounds of a coarray.
- The new intrinsic functions LEADZ and TRAILZ return the number of leading and trailing zero bits in an integer.
- The new intrinsic functions MASKL and MASKR return simple left and right justified masks.
- A BACK= argument has been added to the intrinsic functions MAXLOC and MINLOC.
- The new intrinsic function MERGE_BITS performs a bitwise merge using a mask.
- The new intrinsic function NORM2 calculates the L_2 norm of an array.
- The new intrinsic function NUM_IMAGES returns the number of images.
- The new intrinsic function PARITY reduces an array with the .NEQV. operation.
- The new intrinsic functions POPCNT and POPPAR return the number of 1 bits of an integer and its parity.
- A RADIX= argument has been added to the intrinsic function SELECTED_REAL_KIND.
- The new intrinsic functions SHIFTA, SHIFTL and SHIFTR perform shift operations.
- The new intrinsic function STORAGE_SIZE returns the size of an array element in bits.
- The new intrinsic function THIS_IMAGE returns the index of this image or cosubscripts for it.

- Intrinsic modules:
  The functions COMPILER_VERSION and COMPILER_OPTIONS in the intrinsic module ISO_FORTRAN_ENV return information about the program translation phase. Named constants for selecting kind values have been added to the intrinsic module ISO_FORTRAN_ENV. The function C_SIZEOF in the intrinsic module ISO_C_BINDING returns the size of an array element in bytes. A RADIX= argument has been added to the function IEEE_SELECTED_REAL_KIND in the intrinsic module IEEE_ARITHMETIC.
- Programs and procedures:
  An empty CONTAINS section is allowed. An internal procedure can be used as an actual argument or procedure pointer target. ALLOCATABLE and POINTER attributes are used in generic resolution. Procedureness of a dummy argument is used in generic resolution. An actual argument with the TARGET attribute can correspond to a dummy pointer. A null pointer or unallocated allocatable can be used to denote an absent nonallocatable nonpointer optional argument. An impure elemental procedure processes array arguments in array element order. The FUNCTION and SUBROUTINE keywords can be omitted from the END statement for a module or internal subprogram. A line in the program is permitted to begin with a semicolon.

3 Additionally, the ENTRY feature present in FORTRAN 77 onwards is now deemed to be obsolescent by this part of ISO/IEC 1539.

4 This part of ISO/IEC 1539 is organized in 16 clauses, dealing with 8 conceptual areas. These 8 areas, and the clauses in which they are treated, are:

High/low level concepts	Clauses 1, 2, 3
Data concepts	Clauses 4, 5, 6
Computations	Clauses 7, 13, 14
Execution control	Clause 8
Input/output	Clauses 9, 10
Program units	Clauses 11, 12
Interoperability with C	Clause 15
Scoping and association rules	Clause 16

5 It also contains the following nonnormative material:

Processor dependencies	Annex A
Deleted and obsolescent features	Annex B
Extended notes	Annex C
Syntax rules	Annex D
Index	Index

Information technology — Programming languages — Fortran —

Part 1:
Base language

1 Overview

1.1 Scope

1 This part of ISO/IEC 1539 specifies the form and establishes the interpretation of programs expressed in the base Fortran language. The purpose of this part of ISO/IEC 1539 is to promote portability, reliability, maintainability, and efficient execution of Fortran programs for use on a variety of computing systems.

2 This part of ISO/IEC 1539 specifies

- the forms that a program written in the Fortran language may take,
- the rules for interpreting the meaning of a program and its data,
- the form of the input data to be processed by such a program, and
- the form of the output data resulting from the use of such a program.

3 Except where stated otherwise, requirements and prohibitions specified by this part of ISO/IEC 1539 apply to programs rather than processors.

4 This part of ISO/IEC 1539 does not specify

- the mechanism by which programs are transformed for use on computing systems,
- the operations required for setup and control of the use of programs on computing systems,
- the method of transcription of programs or their input or output data to or from a storage medium,
- the program and processor behavior when this part of ISO/IEC 1539 fails to establish an interpretation except for the processor detection and reporting requirements in items (2) to (8) of 1.5,
- the maximum number of images, or the size or complexity of a program and its data that will exceed the capacity of any particular computing system or the capability of a particular processor,
- the mechanism for determining the number of images of a program,
- the physical properties of an image or the relationship between images and the computational elements of a computing system,
- the physical properties of the representation of quantities and the method of rounding, approximating, or computing numeric values on a particular processor, except by reference to the IEEE International Standard under conditions specified in Clause 14,
- the physical properties of input/output records, files, and units, or
- the physical properties and implementation of storage.

1.2 Normative references

1 The following referenced standards are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 646:1991 (International Reference Version), *Information technology—ISO 7-bit coded character set for information interchange*

ISO/IEC 9899:1999, *Programming languages—C*

ISO/IEC 10646, *Information technology—Universal Multiple-Octet Coded Character Set (UCS)*

IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems*

