

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

Withhold

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



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

Withdrawn



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2010

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Foreword	xiv
Introduction	xv
1 Overview	1
1.1 Scope	1
1.2 Normative references	1
1.3 Terms and definitions	2
1.4 Notation, symbols and abbreviated terms	21
1.4.1 Syntax rules	21
1.4.2 Constraints	22
1.4.3 Assumed syntax rules	22
1.4.4 Syntax conventions and characteristics	22
1.4.5 Text conventions	23
1.5 Conformance	23
1.6 Compatibility	24
1.6.1 New intrinsic procedures	24
1.6.2 Fortran 2003 compatibility	24
1.6.3 Fortran 95 compatibility	24
1.6.4 Fortran 90 compatibility	24
1.6.5 FORTRAN 77 compatibility	25
1.7 Deleted and obsolescent features	25
1.7.1 General	25
1.7.2 Nature of deleted features	25
1.7.3 Nature of obsolescent features	26
2 Fortran concepts	27
2.1 High level syntax	27
2.2 Program unit concepts	30
2.2.1 Program units and scoping units	30
2.2.2 Program	30
2.2.3 Procedure	30
2.2.4 Module	31
2.2.5 Submodule	31
2.3 Execution concepts	31
2.3.1 Statement classification	31
2.3.2 Statement order	31
2.3.3 The END statement	32
2.3.4 Program execution	32
2.3.5 Execution sequence	33
2.4 Data concepts	34
2.4.1 Type	34
2.4.2 Data value	34
2.4.3 Data entity	34
2.4.4 Definition of objects and pointers	36
2.4.5 Reference	36
2.4.6 Array	36
2.4.7 Coarray	37

2.4.8	Pointer	37
2.4.9	Allocatable variables	37
2.4.10	Storage	38
2.5	Fundamental concepts	38
2.5.1	Names and designators	38
2.5.2	Statement keyword	38
2.5.3	Other keywords	38
2.5.4	Association	38
2.5.5	Intrinsic	38
2.5.6	Operator	39
2.5.7	Companion processors	39
3	Lexical tokens and source form	41
3.1	Processor character set	41
3.1.1	Characters	41
3.1.2	Letters	41
3.1.3	Digits	41
3.1.4	Underscore	41
3.1.5	Special characters	41
3.1.6	Other characters	42
3.2	Low-level syntax	42
3.2.1	Tokens	42
3.2.2	Names	42
3.2.3	Constants	43
3.2.4	Operators	43
3.2.5	Statement labels	44
3.2.6	Delimiters	45
3.3	Source form	45
3.3.1	Program units, statements, and lines	45
3.3.2	Free source form	45
3.3.3	Fixed source form	47
3.4	Including source text	47
4	Types	49
4.1	The concept of type	49
4.1.1	General	49
4.1.2	Set of values	49
4.1.3	Constants	49
4.1.4	Operations	49
4.2	Type parameters	49
4.3	Relationship of types and values to objects	51
4.3.1	Type specifiers and type compatibility	51
4.4	Intrinsic types	52
4.4.1	Classification and specification	52
4.4.2	Numeric intrinsic types	53
4.4.3	Character type	56
4.4.4	Logical type	60
4.5	Derived types	60
4.5.1	Derived type concepts	60
4.5.2	Derived-type definition	61
4.5.3	Derived-type parameters	64
4.5.4	Components	66
4.5.5	Type-bound procedures	73
4.5.6	Final subroutines	75
4.5.7	Type extension	77
4.5.8	Derived-type values	79

4.5.9	Derived-type specifier	79
4.5.10	Construction of derived-type values	80
4.5.11	Derived-type operations and assignment	82
4.6	Enumerations and enumerators	82
4.7	Binary, octal, and hexadecimal literal constants	83
4.8	Construction of array values	84
5	Attribute declarations and specifications	87
5.1	General	87
5.2	Type declaration statements	87
5.2.1	Syntax	87
5.2.2	Automatic data objects	88
5.2.3	Initialization	89
5.2.4	Examples of type declaration statements	89
5.3	Attributes	89
5.3.1	Constraints	89
5.3.2	Accessibility attribute	89
5.3.3	ALLOCATABLE attribute	90
5.3.4	ASYNCHRONOUS attribute	90
5.3.5	BIND attribute for data entities	90
5.3.6	CODIMENSION attribute	91
5.3.7	CONTIGUOUS attribute	93
5.3.8	DIMENSION attribute	94
5.3.9	EXTERNAL attribute	96
5.3.10	INTENT attribute	97
5.3.11	INTRINSIC attribute	98
5.3.12	OPTIONAL attribute	99
5.3.13	PARAMETER attribute	99
5.3.14	POINTER attribute	99
5.3.15	PROTECTED attribute	100
5.3.16	SAVE attribute	100
5.3.17	TARGET attribute	101
5.3.18	VALUE attribute	101
5.3.19	VOLATILE attribute	102
5.4	Attribute specification statements	102
5.4.1	Accessibility statement	102
5.4.2	ALLOCATABLE statement	103
5.4.3	ASYNCHRONOUS statement	103
5.4.4	BIND statement	103
5.4.5	CODIMENSION statement	103
5.4.6	CONTIGUOUS statement	104
5.4.7	DATA statement	104
5.4.8	DIMENSION statement	106
5.4.9	INTENT statement	106
5.4.10	OPTIONAL statement	107
5.4.11	PARAMETER statement	107
5.4.12	POINTER statement	107
5.4.13	PROTECTED statement	107
5.4.14	SAVE statement	108
5.4.15	TARGET statement	108
5.4.16	VALUE statement	108
5.4.17	VOLATILE statement	108
5.5	IMPLICIT statement	108
5.6	NAMELIST statement	111
5.7	Storage association of data objects	112
5.7.1	EQUIVALENCE statement	112

5.7.2	COMMON statement	114
5.7.3	Restrictions on common and equivalence	116
6	Use of data objects	117
6.1	Designator	117
6.2	Variable	117
6.3	Constants	118
6.4	Scalars	118
6.4.1	Substrings	118
6.4.2	Structure components	118
6.4.3	Coindexed named objects	120
6.4.4	Complex parts	120
6.4.5	Type parameter inquiry	120
6.5	Arrays	121
6.5.1	Order of reference	121
6.5.2	Whole arrays	121
6.5.3	Array elements and array sections	121
6.5.4	Simply contiguous array designators	124
6.6	Image selectors	125
6.7	Dynamic association	126
6.7.1	ALLOCATE statement	126
6.7.2	NULLIFY statement	129
6.7.3	DEALLOCATE statement	130
6.7.4	STAT= specifier	132
6.7.5	ERRMSG= specifier	132
7	Expressions and assignment	133
7.1	Expressions	133
7.1.1	General	133
7.1.2	Form of an expression	133
7.1.3	Precedence of operators	137
7.1.4	Evaluation of operations	139
7.1.5	Intrinsic operations	139
7.1.6	Defined operations	146
7.1.7	Evaluation of operands	147
7.1.8	Integrity of parentheses	148
7.1.9	Type, type parameters, and shape of an expression	148
7.1.10	Conformability rules for elemental operations	149
7.1.11	Specification expression	150
7.1.12	Constant expression	151
7.2	Assignment	153
7.2.1	Assignment statement	153
7.2.2	Pointer assignment	157
7.2.3	Masked array assignment – WHERE	161
7.2.4	FORALL	163
8	Execution control	169
8.1	Executable constructs containing blocks	169
8.1.1	General	169
8.1.2	Rules governing blocks	169
8.1.3	ASSOCIATE construct	170
8.1.4	BLOCK construct	171
8.1.5	CRITICAL construct	173
8.1.6	DO construct	174
8.1.7	IF construct and statement	179
8.1.8	SELECT CASE construct	181

8.1.9	SELECT TYPE construct	184
8.1.10	EXIT statement	186
8.2	Branching	186
8.2.1	Branch concepts	186
8.2.2	GO TO statement	187
8.2.3	Computed GO TO statement	187
8.2.4	Arithmetic IF statement	187
8.3	CONTINUE statement	187
8.4	STOP and ERROR STOP statements	187
8.5	Image execution control	188
8.5.1	Image control statements	188
8.5.2	Segments	189
8.5.3	SYNC ALL statement	190
8.5.4	SYNC IMAGES statement	190
8.5.5	SYNC MEMORY statement	192
8.5.6	LOCK and UNLOCK statements	193
8.5.7	STAT= and ERRMSG= specifiers in image control statements	195
9	Input/output statements	197
9.1	Input/output concepts	197
9.2	Records	197
9.2.1	General	197
9.2.2	Formatted record	197
9.2.3	Unformatted record	197
9.2.4	Endfile record	198
9.3	External files	198
9.3.1	Basic concepts	198
9.3.2	File existence	198
9.3.3	File access	199
9.3.4	File position	201
9.3.5	File storage units	202
9.4	Internal files	203
9.5	File connection	203
9.5.1	Referring to a file	203
9.5.2	Connection modes	204
9.5.3	Unit existence	205
9.5.4	Connection of a file to a unit	205
9.5.5	Preconnection	206
9.5.6	OPEN statement	206
9.5.7	CLOSE statement	210
9.6	Data transfer statements	211
9.6.1	General	211
9.6.2	Control information list	212
9.6.3	Data transfer input/output list	217
9.6.4	Execution of a data transfer input/output statement	219
9.6.5	Termination of data transfer statements	229
9.7	Waiting on pending data transfer	230
9.7.1	Wait operation	230
9.7.2	WAIT statement	230
9.8	File positioning statements	231
9.8.1	Syntax	231
9.8.2	BACKSPACE statement	232
9.8.3	ENDFILE statement	232
9.8.4	REWIND statement	232
9.9	FLUSH statement	233
9.10	File inquiry statement	233

9.10.1	Forms of the INQUIRE statement	233
9.10.2	Inquiry specifiers	234
9.10.3	Inquire by output list	240
9.11	Error, end-of-record, and end-of-file conditions	240
9.11.1	General	240
9.11.2	Error conditions and the ERR= specifier	240
9.11.3	End-of-file condition and the END= specifier	241
9.11.4	End-of-record condition and the EOR= specifier	241
9.11.5	IOSTAT= specifier	242
9.11.6	IOMSG= specifier	242
9.12	Restrictions on input/output statements	242
10	Input/output editing	245
10.1	Format specifications	245
10.2	Explicit format specification methods	245
10.2.1	FORMAT statement	245
10.2.2	Character format specification	245
10.3	Form of a format item list	246
10.3.1	Syntax	246
10.3.2	Edit descriptors	246
10.3.3	Fields	248
10.4	Interaction between input/output list and format	248
10.5	Positioning by format control	250
10.6	Decimal symbol	250
10.7	Data edit descriptors	250
10.7.1	General	250
10.7.2	Numeric editing	251
10.7.3	Logical editing	256
10.7.4	Character editing	257
10.7.5	Generalized editing	257
10.7.6	User-defined derived-type editing	259
10.8	Control edit descriptors	259
10.8.1	Position editing	259
10.8.2	Slash editing	260
10.8.3	Colon editing	260
10.8.4	SS, SP, and S editing	261
10.8.5	P editing	261
10.8.6	BN and BZ editing	261
10.8.7	RU, RD, RZ, RN, RC, and RP editing	261
10.8.8	DC and DP editing	262
10.9	Character string edit descriptors	262
10.10	List-directed formatting	262
10.10.1	General	262
10.10.2	Values and value separators	262
10.10.3	List-directed input	263
10.10.4	List-directed output	265
10.11	Namelist formatting	266
10.11.1	General	266
10.11.2	Name-value subsequences	266
10.11.3	Namelist input	267
10.11.4	Namelist output	270
11	Program units	271
11.1	Main program	271
11.2	Modules	271
11.2.1	General	271

11.2.2	The USE statement and use association	272
11.2.3	Submodules	275
11.3	Block data program units	275
12	Procedures	277
12.1	Concepts	277
12.2	Procedure classifications	277
12.2.1	Procedure classification by reference	277
12.2.2	Procedure classification by means of definition	277
12.3	Characteristics	278
12.3.1	Characteristics of procedures	278
12.3.2	Characteristics of dummy arguments	278
12.3.3	Characteristics of function results	278
12.4	Procedure interface	279
12.4.1	Interface and abstract interface	279
12.4.2	Implicit and explicit interfaces	279
12.4.3	Specification of the procedure interface	280
12.5	Procedure reference	289
12.5.1	Syntax of a procedure reference	289
12.5.2	Actual arguments, dummy arguments, and argument association	291
12.5.3	Function reference	302
12.5.4	Subroutine reference	302
12.5.5	Resolving named procedure references	302
12.5.6	Resolving type-bound procedure references	305
12.6	Procedure definition	305
12.6.1	Intrinsic procedure definition	305
12.6.2	Procedures defined by subprograms	305
12.6.3	Definition and invocation of procedures by means other than Fortran	311
12.6.4	Statement function	311
12.7	Pure procedures	312
12.8	Elemental procedures	313
12.8.1	Elemental procedure declaration and interface	313
12.8.2	Elemental function actual arguments and results	314
12.8.3	Elemental subroutine actual arguments	314
13	Intrinsic procedures and modules	315
13.1	Classes of intrinsic procedures	315
13.2	Arguments to intrinsic procedures	315
13.2.1	General rules	315
13.2.2	The shape of array arguments	316
13.2.3	Mask arguments	316
13.2.4	DIM arguments and reduction functions	316
13.3	Bit model	317
13.3.1	General	317
13.3.2	Bit sequence comparisons	317
13.3.3	Bit sequences as arguments to INT and REAL	317
13.4	Numeric models	318
13.5	Standard generic intrinsic procedures	318
13.6	Specific names for standard intrinsic functions	323
13.7	Specifications of the standard intrinsic procedures	325
13.7.1	General	325
13.8	Standard modules	396
13.8.1	General	396
13.8.2	The ISO_FORTRAN_ENV intrinsic module	397
14	Exceptions and IEEE arithmetic	401

14.1	General	401
14.2	Derived types and constants defined in the modules	402
14.3	The exceptions	403
14.4	The rounding modes	404
14.5	Underflow mode	405
14.6	Halting	405
14.7	The floating-point status	405
14.8	Exceptional values	405
14.9	IEEE arithmetic	406
14.10	Summary of the procedures	407
14.11	Specifications of the procedures	408
14.11.1	General	408
14.12	Examples	422
15	Interoperability with C	425
15.1	General	425
15.2	The ISO_C_BINDING intrinsic module	425
15.2.1	Summary of contents	425
15.2.2	Named constants and derived types in the module	425
15.2.3	Procedures in the module	426
15.3	Interoperability between Fortran and C entities	429
15.3.1	General	429
15.3.2	Interoperability of intrinsic types	429
15.3.3	Interoperability with C pointer types	431
15.3.4	Interoperability of derived types and C struct types	431
15.3.5	Interoperability of scalar variables	432
15.3.6	Interoperability of array variables	432
15.3.7	Interoperability of procedures and procedure interfaces	433
15.4	Interoperation with C global variables	435
15.4.1	General	435
15.4.2	Binding labels for common blocks and variables	436
15.5	Interoperation with C functions	436
15.5.1	Definition and reference of interoperable procedures	436
15.5.2	Binding labels for procedures	437
15.5.3	Exceptions and IEEE arithmetic procedures	437
16	Scope, association, and definition	439
16.1	Scopes, identifiers, and entities	439
16.2	Global identifiers	439
16.3	Local identifiers	440
16.3.1	Classes of local identifiers	440
16.3.2	Local identifiers that are the same as common block names	441
16.3.3	Function results	441
16.3.4	Components, type parameters, and bindings	441
16.3.5	Argument keywords	442
16.4	Statement and construct entities	442
16.5	Association	443
16.5.1	Name association	443
16.5.2	Pointer association	446
16.5.3	Storage association	449
16.5.4	Inheritance association	451
16.5.5	Establishing associations	451
16.6	Definition and undefinition of variables	452
16.6.1	Definition of objects and subobjects	452
16.6.2	Variables that are always defined	452
16.6.3	Variables that are initially defined	453

16.6.4	Variables that are initially undefined	453
16.6.5	Events that cause variables to become defined	453
16.6.6	Events that cause variables to become undefined	455
16.6.7	Variable definition context	456
16.6.8	Pointer association context	457
Annex A	(informative) Processor Dependencies	459
A.1	Unspecified Items	459
A.2	Processor Dependencies	459
Annex B	(informative) Deleted and obsolescent features	463
B.1	Deleted features	463
B.2	Obsolescent features	464
B.2.1	General	464
B.2.2	Alternate return	464
B.2.3	Computed GO TO statement	464
B.2.4	Statement functions	464
B.2.5	DATA statements among executables	465
B.2.6	Assumed character length functions	465
B.2.7	Fixed form source	465
B.2.8	CHARACTER* form of CHARACTER declaration	465
B.2.9	ENTRY statements	465
Annex C	(informative) Extended notes	467
C.1	Clause 4 notes	467
C.1.1	Selection of the approximation methods (4.4.2.3)	467
C.1.2	Type extension and component accessibility (4.5.2.2, 4.5.4)	467
C.1.3	Generic type-bound procedures (4.5.5)	468
C.1.4	Abstract types (4.5.7.1)	469
C.1.5	Pointers (4.5.4.4, 5.3.14)	470
C.1.6	Structure constructors and generic names (4.5.10)	471
C.1.7	Final subroutines (4.5.6, 4.5.6.2, 4.5.6.3, 4.5.6.4)	473
C.2	Clause 5 notes	474
C.2.1	The POINTER attribute (5.3.14)	474
C.2.2	The TARGET attribute (5.3.17)	475
C.2.3	The VOLATILE attribute (5.3.19)	476
C.3	Clause 6 notes	476
C.3.1	Structure components (6.4.2)	476
C.3.2	Allocation with dynamic type (6.7.1)	478
C.3.3	Pointer allocation and association (6.7.1, 16.5.2)	478
C.4	Clause 7 notes	479
C.4.1	Character assignment (7.2.1.3)	479
C.4.2	Evaluation of function references (7.1.7)	479
C.4.3	Pointers in expressions (7.1.9.2)	480
C.4.4	Pointers in variable-definition contexts (7.2.1.3, 16.6.7)	480
C.4.5	Examples of FORALL constructs (7.2.4)	480
C.4.6	Examples of FORALL statements (7.2.4.3)	482
C.5	Clause 8 notes	483
C.5.1	The SELECT CASE construct (8.1.8)	483
C.5.2	Loop control (8.1.6)	483
C.5.3	Examples of DO constructs (8.1.6)	483
C.5.4	Examples of invalid DO constructs (8.1.6)	485
C.6	Clause 9 notes	486
C.6.1	External files (9.3)	486
C.6.2	Nonadvancing input/output (9.3.4.2)	487
C.6.3	OPEN statement (9.5.6)	488

C.6.4	Connection properties (9.5.4)	490
C.6.5	CLOSE statement (9.5.7)	490
C.6.6	Asynchronous input/output (9.6.2.5)	490
C.7	Clause 10 notes	492
C.7.1	Number of records (10.4, 10.5, 10.8.2)	492
C.7.2	List-directed input (10.10.3)	492
C.8	Clause 11 notes	493
C.8.1	Main program and block data program unit (11.1, 11.3)	493
C.8.2	Dependent compilation (11.2)	493
C.8.3	Examples of the use of modules (11.2.1)	495
C.8.4	Modules with submodules (11.2.3)	501
C.9	Clause 12 notes	505
C.9.1	Portability problems with external procedures (12.4.3.5)	505
C.9.2	Procedures defined by means other than Fortran (12.6.3)	505
C.9.3	Abstract interfaces (12.4) and procedure pointer components (4.5)	505
C.9.4	Pointers and targets as arguments (12.5.2.4, 12.5.2.6, 12.5.2.7)	507
C.9.5	Polymorphic Argument Association (12.5.2.9)	509
C.9.6	Rules ensuring unambiguous generics (12.4.3.4.5)	510
C.10	Clause 13 notes	514
C.10.1	Module for THIS_IMAGE and IMAGE_INDEX	514
C.11	Clause 15 notes	515
C.11.1	Runtime environments (15.1)	515
C.11.2	Example of Fortran calling C (15.3)	515
C.11.3	Example of C calling Fortran (15.3)	516
C.11.4	Example of calling C functions with noninteroperable data (15.5)	518
C.11.5	Example of opaque communication between C and Fortran (15.3)	518
C.12	Clause 16 notes	520
C.12.1	Examples of host association (16.5.1.4)	520
C.13	Array feature notes	521
C.13.1	Summary of features (2.4.6)	521
C.13.2	Examples (6.5)	522
C.13.3	FORMula TRANslation and array processing (6.5)	526
C.13.4	Logical queries (13.7.10, 13.7.13, 13.7.41, 13.7.109, 13.7.115 13.7.161)	528
C.13.5	Parallel computations (7.1.2)	528
C.13.6	Example of element-by-element computation (6.5.3)	528
Annex D	(informative) Syntax rules and constraints	531
D.1	Extract of all syntax rules and constraints	531
D.2	Syntax rule cross-reference	572
Index	585

List of Tables

2.1	Requirements on statement ordering	31
2.2	Statements allowed in scoping units	32
3.1	Special characters	42
6.1	Subscript order value	122
7.1	Categories of operations and relative precedence	137
7.2	Type of operands and results for intrinsic operators	140
7.3	Interpretation of the numeric intrinsic operators	141
7.4	Interpretation of the character intrinsic operator //	143
7.5	Interpretation of the logical intrinsic operators	144
7.6	The values of operations involving logical intrinsic operators	144
7.7	Interpretation of the relational intrinsic operators	145
7.8	Type conformance for the intrinsic assignment statement	153
7.9	Numeric conversion and the assignment statement	155
10.1	E and D exponent forms	253
10.2	EN exponent forms	254
10.3	ES exponent forms	255
13.1	Standard generic intrinsic procedure summary	319
13.2	Characteristics of the result of NULL ()	375
14.1	IEEE_ARITHMETIC module procedure summary	407
14.2	IEEE_EXCEPTIONS module procedure summary	408
15.1	Names of C characters with special semantics	426
15.2	Interoperability between Fortran and C types	430

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

Withdrawn

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*

Withdrawn