

Contents

1	Whence Fortran?	1		
1.1	Introduction	1		
1.2	Fortran's early history	2		
1.3	The drive for the Fortran 90 standard	2		
1.4	Language evolution	4		
1.5	Fortran 95	4		
1.6	Fortran 2003	5		
1.7	Fortran 2008	6		
1.8	Fortran 2018	6		
1.9	Fortran 2023	7		
1.10	Conformance	7		
2	Language elements	9		
2.1	Introduction	9		
2.2	Fortran character set	9		
2.3	Tokens	10		
2.4	Source form	11		
2.5	Concept of type	13		
2.6	Literal constants of intrinsic type	13		
2.6.1	The intrinsic types	13		
2.6.2	Integer literal constants	14		
2.6.3	Real literal constants	15		
2.6.4	Complex literal constants	16		
2.6.5	Character literal constants	17		
2.6.6	Logical literal constants	19		
2.6.7	Binary, octal, and hexadecimal constants	19		
2.7	Names	19		
2.8	Scalar variables of intrinsic type	20		
2.9	Derived data types	21		
2.10	Arrays	22		
2.10.1	Array declarations	22		
2.10.2	Array elements and sections	23		
2.10.3	Array constants	25		
2.10.4	Declaring entities of differing shapes	25		
2.11	Coarrays and images	26		
2.12	Character substrings	26		
2.13	Dynamic memory management and aliasing	27		
2.13.1	Allocatable objects	27		
2.13.2	Pointers	28		
2.14	Type extension	30		
2.15	Polymorphic variables	31		
2.16	Objects and subobjects	31		
2.17	Summary	32		
3	Expressions and assignments	35		
3.1	Introduction	35		
3.2	Scalar numeric expressions	36		
3.3	Defined and undefined variables	39		
3.4	Scalar numeric assignment	39		
3.5	Scalar relational operators	40		
3.6	Scalar logical expressions and assignments	41		
3.7	Scalar character expressions and assignments	42		
3.7.1	Concatenation and assignment	42		
3.7.2	ASCII character set	43		
3.7.3	ISO 10646 character set	44		
3.8	Structure constructors	44		
3.9	Scalar defined operators	45		
3.10	Scalar defined assignments	47		
3.11	Array expressions	49		
3.12	Array assignment	50		
3.13	Pointers in expressions and assignments	51		
3.14	The nullify statement	53		
3.15	Summary	53		
4	Control constructs	56		
4.1	Introduction	56		
4.2	The if construct and statement	56		
4.3	The case construct	58		
4.4	The do construct	60		
4.5	Further uses of the exit statement	63		
4.6	Summary	64		
5	Program units and procedures	69		
5.1	Introduction	69		
5.2	Main program	70		
5.3	Program termination	71		
5.4	External subprograms	72		
5.5	Modules	73		

5.6	Internal subprograms	75	6.11.1	Allocatable components of recursive type	116
5.7	Arguments of procedures	76	6.12	Allocatable arrays vs. pointers	118
5.7.1	Argument association	76	6.13	Summary	119
5.7.2	Assumed-shape arrays	78	7	Array features	120
5.7.3	Pointer arguments	79	7.1	Introduction	120
5.7.4	Polymorphic arguments	79	7.2	Zero-sized arrays	120
5.7.5	Restrictions on actual arguments	80	7.3	Automatic objects	121
5.7.6	Arguments with the target attribute	81	7.4	Elemental operations and assignments	122
5.8	The return statement	81	7.5	Array-valued functions	123
5.9	Local variables	81	7.6	The where statement and construct	123
5.10	Argument intent	82	7.7	Mask arrays	127
5.11	Functions	83	7.8	Pure procedures	127
5.11.1	Function results	83	7.9	Elemental procedures	128
5.11.2	Prohibited side-effects	84	7.9.1	Pure elemental procedures	128
5.12	Explicit interfaces	85	7.9.2	Impure elemental procedures	129
5.12.1	Interface blocks	85	7.10	Array elements	130
5.12.2	The import statement	86	7.11	Array subobjects	131
5.13	Procedures as arguments	88	7.12	Arrays of pointers	135
5.14	Keyword and optional arguments	90	7.13	Pointers as aliases	135
5.15	Use and scope of labels	91	7.14	Remapping bounds and rank in pointer assignments	136
5.16	Scope of names	91	7.15	Array constructors	137
5.17	Recursion	94	7.16	The do concurrent construct	138
5.17.1	Direct recursion	94	7.17	Contiguous arrays	142
5.17.2	Indirect recursion	95	7.17.1	The contiguous attribute	142
5.17.3	Non-recursive procedures	96	7.17.2	Simply contiguous array designators	145
5.18	Overloading	96	7.17.3	Automatic pointer targetting	146
5.18.1	Generic interfaces	96	7.18	Assumed rank	147
5.18.2	Distinguishing generic invocations	98	7.19	Summary	149
5.19	Assumed character length	101	8	Specification statements	152
5.20	The subroutine and function statements	103	8.1	Introduction	152
5.21	Requirements on statement ordering	104	8.2	Implicit declarations	152
5.22	Summary	104	8.2.1	Implicit typing	152
6	Allocation of data	106	8.2.2	Requiring explicit procedure and type declarations	153
6.1	Introduction	106	8.3	Named constants	153
6.2	The allocatable attribute	106	8.4	Constant expressions	155
6.3	Deferred type parameters	107	8.5	Initial values for variables	157
6.4	Allocatable scalars	108	8.5.1	Initialization in type declaration statements	157
6.5	The allocate statement	108	8.5.2	The data statement	157
6.6	The deallocate statement	110	8.5.3	Pointer initialization as disassociated	160
6.7	Automatic reallocation in intrinsic assignment	111	8.5.4	Pointer initialization as associated	161
6.8	Transferring an allocation	111	8.5.5	Default initialization of components	161
6.9	Allocatable dummy arguments	113	8.6	Accessibility	162
6.10	Allocatable functions	113	8.6.1	The public and private attributes	162
6.11	Allocatable components	114			

8.6.2	The protected attribute	165	9.9.4	Transformational functions for kind values	201
8.7	Pointer functions denoting variables	165	9.9.5	Checking for unsafe conversions	202
8.8	The pointer, target, and allocatable statements	166	9.10	Bit manipulation procedures	202
8.9	The intent and optional statements	168	9.10.1	Model for bit data	202
8.10	The save attribute	168	9.10.2	Inquiry function	202
8.11	Asynchronous actions	169	9.10.3	Basic elemental functions	203
8.11.1	Introduction	169	9.10.4	Shift operations	204
8.11.2	The asynchronous attribute	170	9.10.5	Elemental subroutine	204
8.12	The block construct	170	9.10.6	Bitwise (unsigned) comparison	205
8.13	The use statement	172	9.10.7	Double-width shifting	205
8.14	Derived-type definitions	175	9.10.8	Bitwise reductions	206
8.15	The type declaration statement	177	9.10.9	Counting bits	206
8.16	Type and type parameter specification	178	9.10.10	Producing bitmasks	206
8.17	Specification expressions	179	9.10.11	Merging bits	206
8.18	Structure constructors	182	9.11	Transfer function	207
8.19	The namelist statement	183	9.12	Vector and matrix multiplication functions	207
8.20	Summary	184	9.13	Transformational functions that reduce arrays	208
9	Intrinsic procedures and modules	188	9.13.1	Single-argument case	208
9.1	Introduction	188	9.13.2	Additional arguments <code>dim</code> and <code>kind</code>	209
9.1.1	Extent and performance of the intrinsic procedures	188	9.13.3	Optional argument <code>mask</code>	210
9.1.2	Keyword calls	188	9.13.4	Generalized array reduction	210
9.1.3	Categories of intrinsic procedures	189	9.14	Array inquiry functions	211
9.1.4	The intrinsic statement	189	9.14.1	Introduction	211
9.1.5	Argument intents	189	9.14.2	Contiguity	211
9.2	Inquiry functions for any type	190	9.14.3	Bounds, shape, and size	211
9.3	Elemental numeric functions	190	9.15	Array construction and manipulation functions	212
9.3.1	Introduction	190	9.15.1	The merge elemental function	212
9.3.2	Elemental functions that may convert	191	9.15.2	Packing and unpacking arrays	212
9.3.3	Elemental functions that do not convert	192	9.15.3	Reshaping an array	213
9.4	Elemental mathematical functions	193	9.15.4	Transformational function for replication	213
9.5	Transformational functions for Bessel functions	195	9.15.5	Array shifting functions	213
9.6	Elemental character and logical functions	196	9.15.6	Matrix transpose	214
9.6.1	Character-integer conversions	196	9.16	Transformational functions for geometric location	214
9.6.2	Lexical comparison functions	196	9.17	Transformational function for disassociated or unallocated entities	215
9.6.3	String-handling elemental functions	197	9.18	Non-elemental intrinsic subroutines	216
9.6.4	Logical conversion	197	9.18.1	Introduction	216
9.7	Non-elemental string-handling functions	198	9.18.2	Real-time clock	216
9.7.1	String-handling inquiry function	198	9.18.3	CPU time	217
9.7.2	String-handling transformational functions	198	9.18.4	Random numbers	217
9.8	Character inquiry function	198	9.18.5	Executing another program	218
9.9	Numeric inquiry and manipulation functions	198	9.19	Access to the computing environment	219
9.9.1	Models for integer and real data	198	9.19.1	Environment variables	219
9.9.2	Numeric inquiry functions	199	9.19.2	Information about the program invocation	220
9.9.3	Elemental functions to manipulate reals	200	9.20	Elemental functions for input/output status testing	221
			9.21	Size of an object in memory	221

9.22	Miscellaneous procedures	222	11.4	Control edit descriptors	260
9.23	Intrinsic modules	222	11.4.1	Introduction	260
9.24	Fortran environment	223	11.4.2	Scale factor	260
9.24.1	Introduction	223	11.4.3	Tabulation and spacing	261
9.24.2	Named constants	223	11.4.4	New records (slash editing)	262
9.24.3	Compilation information	225	11.4.5	Colon editing	262
9.24.4	Names for common kinds	225	11.5	Changeable file connection modes	263
9.24.5	Kind arrays	226	11.5.1	Introduction	263
9.24.6	Derived types for coarray programming	227	11.5.2	Embedded blank interpretation	263
9.25	Summary	227	11.5.3	Input/output rounding mode	264
10	Data transfer	229	11.5.4	Signs on positive values	264
10.1	Introduction	229	11.5.5	Decimal comma for input/output	265
10.2	Number conversion	229	11.5.6	Padding input records	265
10.3	Input/output lists	230	11.5.7	Delimiting character values	266
10.4	Format definition	232	11.6	Defined derived-type input/output	266
10.5	Unit numbers	234	11.7	Recursive input/output	269
10.6	Internal files	235	11.8	Summary	270
10.7	Formatted input	237	12	Operations on external files	272
10.8	Formatted output	239	12.1	Introduction	272
10.9	List-directed input/output	240	12.2	Positioning statements for sequential files	273
10.10	Namelist input/output	242	12.2.1	The backspace statement	273
10.11	Non-advancing input/output	244	12.2.2	The rewind statement	273
10.12	Unformatted input/output	245	12.2.3	The endfile statement	274
10.13	Direct-access files	246	12.2.4	Data transfer statements	274
10.14	UTF-8 files	247	12.3	The flush statement	275
10.15	Asynchronous input/output	248	12.4	The open statement	275
10.16	Stream access files	250	12.5	The close statement	278
10.17	Execution of a data transfer statement	251	12.6	The inquire statement	279
10.18	Summary	252	12.6.1	Introduction	279
11	Edit descriptors	253	12.6.2	Inquire by file or unit	279
11.1	Introduction	253	12.6.3	Inquire by input/output list	282
11.2	Character string edit descriptor	253	12.7	Summary	283
11.3	Data edit descriptors	253	13	Further type parameter features	284
11.3.1	Introduction	253	13.1	Type parameter inquiry	284
11.3.2	Repeat counts	254	13.2	Parameterized derived types	284
11.3.3	Integer formatting	255	13.2.1	Defining a parameterized derived type	285
11.3.4	Real formatting with e, en, es, and f edit descriptors	255	13.2.2	Assumed and deferred type parameters	286
11.3.5	Hexadecimal significand input/output	257	13.2.3	Default type parameter values	286
11.3.6	Complex formatting	258	13.2.4	Derived type parameter inquiry	287
11.3.7	Logical formatting	258	13.2.5	Structure constructor	287
11.3.8	Character formatting	258	14	Abstract interfaces and procedure pointers	289
11.3.9	General formatting	259	14.1	Abstract interfaces	289
11.3.10	Derived-type formatting	260			

14.2	Procedure pointers	290	16.3	Submodules of submodules	326
14.2.1	Introduction	290	16.4	Submodule entities	327
14.2.2	Named procedure pointers	290	16.5	Submodules and use association	327
14.2.3	Procedure pointer components	291	16.6	The advantages of submodules	328
14.2.4	The <code>pass</code> attribute	292			
14.2.5	Internal procedures as targets of a procedure pointer	293	17	Coarrays	329
15	Object-oriented programming	294	17.1	Introduction	329
15.1	Introduction	294	17.2	Referencing images	330
15.2	Type extension	294	17.3	The properties of coarrays	331
15.2.1	Introduction	294	17.4	Accessing coarrays	332
15.2.2	Parent component	295	17.5	The <code>sync all</code> statement	333
15.2.3	Extension without adding components	296	17.6	Allocatable coarrays and coarray components	335
15.2.4	Type extension and type parameters	296	17.7	Coarrays with allocatable or pointer components	337
15.3	Polymorphic entities	296	17.7.1	Introduction	337
15.3.1	Introduction	296	17.7.2	Data components	337
15.3.2	Establishing the dynamic type	297	17.7.3	Procedure pointer components	338
15.3.3	Limitations on the use of a polymorphic variable	298	17.8	Coarrays in procedures	339
15.3.4	Polymorphic arrays and scalars	298	17.9	Asynchronous attribute	341
15.3.5	Unlimited polymorphic entities	299	17.10	Interoperability	341
15.3.6	Polymorphic entities and generic resolution	300	17.11	Execution segments	341
15.4	Typed and sourced allocation	301	17.12	The <code>sync images</code> statement	342
15.4.1	Introduction	301	17.13	The <code>lock</code> and <code>unlock</code> statements	343
15.4.2	Typed allocation and deferred type parameters	302	17.14	Critical sections	345
15.4.3	Polymorphic variables and typed allocation	302	17.15	Events	346
15.4.4	Sourced allocation	303	17.16	Derived types for locks, events, and teams	347
15.5	Assignment for allocatable polymorphic variables	304	17.17	The image control statements	348
15.6	The <code>associate</code> construct	305	17.18	Error termination	348
15.7	The <code>select type</code> construct	306	17.19	Normal termination	349
15.8	Type-bound procedures	308	17.20	Input/output	349
15.8.1	Introduction	308	17.21	Intrinsic procedures	350
15.8.2	Specific type-bound procedures	309	17.21.1	Introduction	350
15.8.3	Generic type-bound procedures	310	17.21.2	Inquiry functions	351
15.8.4	Type extension and type-bound procedures	313	17.21.3	Transformational functions	351
15.9	Design for overriding	314	17.22	Collective subroutines	352
15.10	Deferred bindings and abstract types	316	17.23	Atomic subroutines	353
15.11	Finalization	317	17.24	Image failure	355
15.11.1	Introduction	317	17.25	Diagnosing the state of a parallel computation	356
15.11.2	Type extension and final subroutines	318	17.25.1	Detecting failed and stopped images	356
15.12	Procedure encapsulation example	320	17.25.2	Coping with stopped or failed images	357
15.13	Type inquiry functions	321			
16	Submodules	325	18	Coarray teams	360
16.1	Introduction	325	18.1	Teams	360
16.2	Separate module procedures	325	18.2	The <code>form team</code> statement	361
			18.3	The <code>change team</code> construct	362
			18.4	Coarrays allocated in teams	362

18.5	The sync team statement	363
18.6	Image selectors and teams	363
18.7	Procedure calls and teams	363
18.8	Intrinsic functions	364
18.8.1	Intrinsic functions <code>get_team</code> and <code>team_number</code>	364
18.8.2	Intrinsic function <code>image_index</code>	365
18.8.3	Intrinsic function <code>num_images</code>	366
18.8.4	Intrinsic function <code>this_image</code>	366
18.9	Detecting failed and stopped images	366
18.10	Recovering from image failure	367
19	Floating-point exception handling	371
19.1	Introduction	371
19.2	The IEEE standard	371
19.3	Access to the features	373
19.4	The Fortran flags	375
19.5	The floating-pointing modes	376
19.6	The <code>ieee_exceptions</code> module	377
19.6.1	Introduction	377
19.6.2	Derived types for floating-point flags, modes, and status	377
19.6.3	Inquiring about support of IEEE exceptions	378
19.6.4	Subroutines for getting and setting the flags and modes	378
19.7	The <code>ieee_arithmetic</code> module	379
19.7.1	Introduction	379
19.7.2	Derived types	380
19.7.3	Inquiring about IEEE arithmetic	381
19.7.4	Elemental functions	382
19.7.5	Non-elemental subroutines	386
19.8	Examples	387
19.8.1	Dot product	387
19.8.2	Calling alternative procedures	388
19.8.3	Calling alternative in-line code	389
19.8.4	Reliable hypotenuse function	389
19.8.5	Access to IEEE arithmetic values	390
20	Basic interoperability with C	393
20.1	Introduction	393
20.2	Interoperability of intrinsic types	393
20.3	Interoperability with C pointer types	395
20.4	Interoperability of derived types	397
20.5	Shape and character length disagreement	397
20.6	Interoperability of variables	400
20.7	Function <code>c_sizeof</code>	400
20.8	The value attribute	401
20.9	Interoperability of procedures	402

20.10	Interoperability of global data	404
20.11	Invoking a C function from Fortran	404
20.12	Invoking Fortran from C	406
20.13	Enumerations	408
20.14	Optional arguments	408
20.15	Assumed-type dummy arguments	409
20.16	Assumed character length	410

21 Interoperating with C using descriptors **412**

21.1	Introduction	412
21.2	C descriptors	412
21.2.1	Introduction	412
21.2.2	Standard members	413
21.2.3	Argument classification (attribute codes)	414
21.2.4	Argument data type	414
21.2.5	Array layout information	414
21.3	Accessing Fortran objects	415
21.3.1	Traversing contiguous Fortran arrays	415
21.3.2	Generic programming with assumed type	417
21.3.3	Traversing discontinuous Fortran arrays	418
21.3.4	Fortran pointer operations	419
21.3.5	Allocatable objects	420
21.3.6	Handling arrays of any rank	422
21.3.7	Accessing individual array elements via a C descriptor	423
21.3.8	Handling errors from CFI functions	426
21.4	Calling Fortran with C descriptors	426
21.4.1	Allocating storage for a C descriptor	426
21.4.2	Establishing a C descriptor	427
21.4.3	Constructing an array section	429
21.4.4	Accessing components	430
21.5	Restrictions	432
21.5.1	Other limitations on C descriptors	432
21.5.2	Lifetimes of C descriptors	432

22 Generic programming **433**

22.1	Introduction	433
22.2	Genericity in type	434
22.2.1	Copying the type and type parameters of another object	434
22.2.2	Type-independent programming	435
22.3	Genericity in rank	437
22.3.1	Rank-independent array declarations	437
22.3.2	Rank-independent allocation	438
22.3.3	Rank-independent array accessing	439
22.4	Future directions	442

23 Other Fortran 2023 enhancements	443
23.1 Introduction	443
23.2 Language elements	443
23.2.1 Longer lines and overall statement length	443
23.2.2 Automatic allocation of lengths of character variables	443
23.2.3 Conditional expressions and arguments	444
23.2.4 More use of binary, octal, and hexadecimal constants	445
23.3 Intrinsic procedures and intrinsic modules	446
23.3.1 Extracting tokens from a string	446
23.3.2 Trigonometric functions that work in degrees	449
23.3.3 Trigonometric functions that work with half revolutions	449
23.3.4 The function <code>selected_logical_kind</code>	450
23.3.5 Changes to <code>system_clock</code>	450
23.3.6 Changes for conformance with new IEEE standard	450
23.3.7 Additional named constants to specify kinds	451
23.4 Interoperability with C	451
23.4.1 Extension to the intrinsic procedure <code>c_f_pointer</code>	451
23.4.2 Procedures for converting between Fortran and C strings	452
23.5 Input/output	454
23.5.1 The <code>at edit</code> descriptor	454
23.5.2 Control over leading zeros in output of real values	454
23.5.3 Namelist with private variables	455
23.6 Coarrays	455
23.6.1 An array or allocatable object may have a coarray component	455
23.6.2 Put with notify	457
23.6.3 Error conditions in collectives	459
23.7 Procedures	459
23.7.1 Simple procedures	459
23.7.2 Reduction specifier for <code>do concurrent</code>	460
23.8 Enumerations	461
23.8.1 Introduction	461
23.8.2 Enumeration types	462
23.8.3 Enum types	464
A Deprecated features	467
A.1 Introduction	467
A.2 The <code>include</code> line	467
A.3 Alternative form of complex constant	468
A.4 Double precision real	468
A.5 Type statement for declaring an entity of intrinsic type	468
A.6 The dimension, codimension, and parameter statements	469
A.7 Sequence types	470
A.8 Storage association	470
A.9 Non-default mapping for implicit typing	471
A.10 Alternative form of relational operator	473

A.11 The <code>do while</code> statement	474
A.12 Control of execution flow by branching	474
A.12.1 The <code>go to</code> statement	474
A.12.2 The <code>continue</code> statement	475
A.12.3 Branching in input/output statements	475
A.13 Implicit interfaces	475
A.14 Denoting an absent non-pointer non-allocatable argument	477
A.15 The <code>volatile</code> attribute	478
A.15.1 Volatile scoping	480
A.15.2 Volatile arguments	481
A.15.3 Volatile coarrays	481
A.16 The <code>sync memory</code> statement	481
A.17 Coarray components of type <code>c_ptr</code> or <code>c_funptr</code>	483
B Obsolescent and deleted features	484
B.1 Obsolescent features	484
B.1.1 Introduction	484
B.1.2 Fixed source form	484
B.1.3 Character length specification with <code>character*</code>	485
B.1.4 data statements among executables	485
B.1.5 Computed <code>go to</code>	485
B.1.6 The equivalence statement	486
B.1.7 The <code>common block</code>	487
B.1.8 The <code>block data</code> program unit	490
B.1.9 Statement functions	490
B.1.10 Assumed character length of function results	492
B.1.11 Alternate return	492
B.1.12 The <code>entry</code> statement	493
B.1.13 The <code>forall</code> statement and construct	495
B.1.14 The labelled <code>do</code> construct	498
B.1.15 Specific names of intrinsic procedures	499
B.2 Deleted features	503
C Significant examples	505
C.1 Introduction	505
C.2 Object-oriented list example	505
C.3 Matrix–vector multiplication	506
C.4 Triangular matrix by vector multiplication	509
C.5 Cholesky factorization	512
D Solutions to exercises	517
Index	528