Modern Fortran Explained

Incorporating Fortran 2018

Michael Metcalf, John Reid, and Malcolm Cohen

OXFORD
UNIVERSITY PRESS

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 3
1.5 Fortran 95 4
1.6 Extensions to Fortran 95 5
1.7 Fortran 2003 5
1.8 Extensions to Fortran 2003 6
1.9 Fortran 2008 6
1.10 Extensions to Fortran 2008 7
1.11 Fortran 2018 7
1.12 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 Integer literal constants 14
  2.6.2 Real literal constants 15
  2.6.3 Complex literal constants 16
  2.6.4 Character literal constants 16
  2.6.5 Logical literal constants 18
  2.6.6 Binary, octal, and hexadecimal constants 19
2.7 Names 19
2.8 Scalar variables of intrinsic type 20
2.9 Derived data types 20
2.10 Arrays of intrinsic type 22
  2.10.1 Declaring entities of differing shapes 25
  2.10.2 Allocatable objects 25

2.11 Character substrings 26
2.12 Pointers 27
2.13 Objects and subobjects 28
2.14 Summary 29

3 Expressions and assignments **33**
3.1 Introduction 33
3.2 Scalar numeric expressions 34
3.3 Defined and undefined variables 37
3.4 Scalar numeric assignment 37
3.5 Scalar relational operators 38
3.6 Scalar logical expressions and assignments 39
3.7 Scalar character expressions and assignments 40
3.7.1 ASCII character set 41
3.7.2 ISO 10646 character set 42
3.8 Structure constructors 42
3.9 Scalar defined operators 43
3.10 Scalar defined assignments 45
3.11 Array expressions 47
3.12 Array assignment 48
3.13 Pointers in expressions and assignments 49
3.14 The nullify statement 51
3.15 Summary 51

4 Control constructs **55**
4.1 Introduction 55
4.2 The if construct and statement 55
4.3 The case construct 57
4.4 The do construct 59
4.5 Exit from nearly any construct 62
4.6 The go to statement 63
4.7 Summary 64

5 Program units and procedures **69**
5.1 Introduction 69
5.2 Main program 70
5.3 The stop statement 71
5.4 External subprograms 72
5.5 Modules 73
5.6 Internal subprograms 75
5.7 Arguments of procedures 76
5.7.1 Assumed-shape arrays 78
5.7.2 Pointer arguments 78
5.7.3 Restrictions on actual arguments 79
5.7.4 Arguments with the target attribute 80

5.8 The return statement 80
5.9 Argument intent 81
5.10 Functions 82
5.10.1 Prohibited side-effects 83
5.11 Explicit and implicit interfaces 83
5.11.1 The import statement 85
5.12 Procedures as arguments 86
5.13 Keyword and optional arguments 88
5.14 Scope of labels 90
5.15 Scope of names 90
5.16 Direct recursion 92
5.17 Indirect recursion 94
5.18 Overloading and generic interfaces 94
5.19 Assumed character length 99
5.20 The subroutine and function statements 99
5.21 Summary 101

6 Allocation of data **105**
6.1 Introduction 105
6.2 The allocatable attribute 105
6.3 Deferred type parameters 106
6.4 Allocatable scalars 106
6.5 The allocate statement 107
6.6 The deallocate statement 108
6.7 Automatic reallocation 109
6.8 Transferring an allocation 110
6.9 Allocatable dummy arguments 111
6.10 Allocatable functions 111
6.11 Allocatable components 112
6.11.1 Allocatable components of recursive type 115
6.12 Allocatable arrays vs. pointers 116
6.13 Summary 117

7 Array features **119**
7.1 Introduction 119
7.2 Zero-sized arrays 119
7.3 Automatic objects 120
7.4 Elemental operations and assignments 121
7.5 Array-valued functions 122
7.6 The where statement and construct 123
7.7 Mask arrays 126
7.8 Pure procedures 126
7.9 Elemental procedures 127
7.10 Impure elemental procedures 128
7.11 Array elements 129

7.12 Array subobjects 130
7.13 Arrays of pointers 133
7.14 Pointers as aliases 134
7.15 Pointer assignment 135
7.16 Array constructors 135
7.17 The do concurrent construct 137
7.18 Performance-oriented features 139
7.18.1 The contiguous attribute 139
7.18.2 Simply contiguous array designators 142
7.18.3 Automatic pointer targetting 144
7.19 Summary 145

8 Specification statements **149**
8.1 Introduction 149
8.2 Implicit typing 149
8.3 Named constants 150
8.4 Constant expressions 152
8.5 Initial values for variables 153
8.5.1 Initialization in type declaration statements 153
8.5.2 The data statement 154
8.5.3 Pointer initialization as disassociated 156
8.5.4 Pointer initialization as associated 157
8.5.5 Default initialization of components 158
8.6 Accessibility 159
8.6.1 The public and private attributes 159
8.6.2 More control of access from a module 161
8.7 Pointer functions denoting variables 161
8.8 The pointer, target, and allocatable statements 163
8.9 The intent and optional statements 163
8.10 The save attribute 164
8.11 Volatility 165
8.11.1 The volatile attribute 165
8.11.2 Volatile scoping 167
8.11.3 Volatile arguments 167
8.12 The asynchronous attribute 168
8.13 The block construct 168
8.14 The use statement 170
8.15 Derived-type definitions 173
8.16 The type declaration statement 176
8.17 Type and type parameter specification 177
8.18 Specification expressions 178
8.19 Structure constructors 181
8.20 The namelist statement 181
8.21 Summary 183

9 Intrinsic procedures and modules **187**

9.1 Introduction 187

9.1.1 Keyword calls 187

9.1.2 Categories of intrinsic procedures 188

9.1.3 The intrinsic statement 188

9.1.4 Argument intents 188

9.2 Inquiry functions for any type 188

9.3 Elemental numeric functions 189

9.3.1 Elemental functions that may convert 189

9.3.2 Elemental functions that do not convert 191

9.4 Elemental mathematical functions 191

9.5 Transformational functions for Bessel functions 194

9.6 Elemental character and logical functions 194

9.6.1 Character–integer conversions 194

9.6.2 Lexical comparison functions 195

9.6.3 String-handling elemental functions 195

9.6.4 Logical conversion 196

9.7 Non-elemental string-handling functions 196

9.7.1 String-handling inquiry function 196

9.7.2 String-handling transformational functions 196

9.8 Character inquiry function 197

9.9 Numeric inquiry and manipulation functions 197

9.9.1 Models for integer and real data 197

9.9.2 Numeric inquiry functions 198

9.9.3 Elemental functions to manipulate reals 199

9.9.4 Transformational functions for kind values 199

9.10 Bit manipulation procedures 200

9.10.1 Model for bit data 200

9.10.2 Inquiry function 200

9.10.3 Basic elemental functions 201

9.10.4 Shift operations 202

9.10.5 Elemental subroutine 202

9.10.6 Bitwise (unsigned) comparison 203

9.10.7 Double-width shifting 203

9.10.8 Bitwise reductions 204

9.10.9 Counting bits 204

9.10.10 Producing bitmasks 204

9.10.11 Merging bits 204

9.11 Transfer function 205

9.12 Vector and matrix multiplication functions 205

9.13 Transformational functions that reduce arrays 206

9.13.1 Single argument case 206

9.13.2 Additional argument dim 207

9.13.3 Optional argument mask 207

9.14 Array inquiry functions 208

9.14.1 Contiguity 208
9.14.2 Bounds, shape, and size 208
9.15 Array construction and manipulation functions 209
9.15.1 The merge elemental function 209
9.15.2 Packing and unpacking arrays 209
9.15.3 Reshaping an array 210
9.15.4 Transformational function for replication 210
9.15.5 Array shifting functions 210
9.15.6 Matrix transpose 211
9.16 Transformational functions for geometric location 211
9.17 Transformational function for disassociated or unallocated 212
9.18 Non-elemental intrinsic subroutines 212
9.18.1 Real-time clock 212
9.18.2 CPU time 213
9.18.3 Random numbers 214
9.18.4 Executing another program 214
9.19 Access to the computing environment 215
9.19.1 Environment variables 215
9.19.2 Information about the program invocation 216
9.20 Elemental functions for I/O status testing 217
9.21 Size of an object in memory 217
9.22 Miscellaneous procedures 218
9.23 Intrinsic modules 218
9.24 Fortran environment 219
9.24.1 Named constants 219
9.24.2 Compilation information 220
9.24.3 Names for common kinds 220
9.24.4 Kind arrays 221
9.24.5 Lock type 222
9.25 Summary 222

10 Data transfer **225**
10.1 Introduction 225
10.2 Number conversion 225
10.3 I/O lists 226
10.4 Format definition 228
10.5 Unit numbers 230
10.6 Internal files 231
10.7 Formatted input 233
10.8 Formatted output 234
10.9 List-directed I/O 236
10.10 Namelist I/O 238
10.11 Non-advancing I/O 239
10.12 Unformatted I/O 240
10.13 Direct-access files 241

10.14 UTF-8 files 242
10.15 Asynchronous input/output 243
10.15.1 Asynchronous execution 243
10.15.2 The asynchronous attribute 245
10.16 Stream access files 246
10.17 Execution of a data transfer statement 247
10.18 Summary 248

11 Edit descriptors **249**
11.1 Introduction 249
11.2 Character string edit descriptor 249
11.3 Data edit descriptors 249
11.3.1 Repeat counts 250
11.3.2 Integer formatting 251
11.3.3 Real formatting 251
11.3.4 Complex formatting 253
11.3.5 Logical formatting 253
11.3.6 Character formatting 253
11.3.7 General formatting 254
11.3.8 Derived type formatting 255
11.4 Control edit descriptors 255
11.4.1 Scale factor 255
11.4.2 Tabulation and spacing 256
11.4.3 New records (slash editing) 257
11.4.4 Colon editing 257
11.5 Changeable file connection modes 258
11.5.1 Embedded blank interpretation 258
11.5.2 Input/output rounding mode 259
11.5.3 Signs on positive values 259
11.5.4 Decimal comma for input/output 260
11.6 Defined derived-type input/output 260
11.7 Recursive input/output 264
11.8 Summary 265

12 Operations on external files **267**
12.1 Introduction 267
12.2 Positioning statements for sequential files 268
12.2.1 The backspace statement 268
12.2.2 The rewind statement 268
12.2.3 The endfile statement 269
12.2.4 Data transfer statements 269
12.3 The flush statement 269
12.4 The open statement 270
12.5 The close statement 273
12.6 The inquire statement 274

12.7 Summary 278

13 Advanced type parameter features **279**
13.1 Type parameter inquiry 279
13.2 Parameterized derived types 279
13.2.1 Defining a parameterized derived type 280
13.2.2 Assumed and deferred type parameters 281
13.2.3 Default type parameter values 281
13.2.4 Derived type parameter inquiry 282
13.2.5 Structure constructor 282

14 Procedure pointers **285**
14.1 Abstract interfaces 285
14.2 Procedure pointers 287
14.2.1 Named procedure pointers 287
14.2.2 Procedure pointer components 287
14.2.3 The pass attribute 288
14.2.4 Internal procedures as targets of a procedure pointer 289

15 Object-oriented programming **291**
15.1 Introduction 291
15.2 Type extension 291
15.2.1 Type extension and type parameters 293
15.3 Polymorphic entities 293
15.3.1 Introduction to polymorphic entities 293
15.3.2 Establishing the dynamic type 294
15.3.3 Limitations on the use of a polymorphic variable 295
15.3.4 Polymorphic arrays and scalars 295
15.3.5 Unlimited polymorphic entities 295
15.3.6 Polymorphic entities and generic resolution 296
15.4 Typed and sourced allocation 297
15.4.1 Introduction 297
15.4.2 Typed allocation and deferred type parameters 297
15.4.3 Polymorphic variables and typed allocation 298
15.4.4 Sourced allocation 298
15.5 Assignment for allocatable polymorphic variables 300
15.6 The associate construct 300
15.7 The select type construct 302
15.8 Type-bound procedures 303
15.8.1 Specific type-bound procedures 305
15.8.2 Generic type-bound procedures 306
15.8.3 Type extension and type-bound procedures 308
15.9 Design for overriding 310
15.10 Deferred bindings and abstract types 312
15.11 Finalization 313

15.11.1 Type extension and final subroutines . . . 315
15.12 Procedure encapsulation example . . . 315
15.13 Type inquiry functions . . . 319

16 Submodules **321**
16.1 Introduction . . . 321
16.2 Separate module procedures . . . 321
16.3 Submodules of submodules . . . 323
16.4 Submodule entities . . . 323
16.5 Submodules and use association . . . 323
16.6 The advantages of submodules . . . 324

17 Coarrays **325**
17.1 Introduction . . . 325
17.2 Referencing images . . . 326
17.3 The properties of coarrays . . . 327
17.4 Accessing coarrays . . . 328
17.5 The sync all statement . . . 329
17.6 Allocatable coarrays . . . 330
17.7 Coarrays with allocatable or pointer components . . . 331
17.7.1 Data components . . . 332
17.7.2 Procedure pointer components . . . 333
17.8 Coarray components . . . 333
17.9 Coarrays in procedures . . . 333
17.10 References to polymorphic subobjects . . . 335
17.11 Volatile and asynchronous attributes . . . 336
17.12 Interoperability . . . 336
17.13 Synchronization . . . 336
17.13.1 Execution segments . . . 336
17.13.2 The sync images statement . . . 337
17.13.3 The lock and unlock statements . . . 338
17.13.4 Critical sections . . . 340
17.13.5 Atomic subroutines and the sync memory statement . . . 341
17.13.6 The stat= and errmsg= specifiers in synchronization statements . . . 341
17.13.7 The image control statements . . . 342
17.14 Program termination . . . 342
17.15 Input/output . . . 344
17.16 Intrinsic procedures . . . 344
17.16.1 Inquiry functions . . . 344
17.16.2 Transformational functions . . . 345
17.17 Arrays of different sizes on different images . . . 345

18 Floating-point exception handling **347**
18.1 Introduction . . . 347
18.2 The IEEE standard . . . 347

18.3 Access to the features 349
18.4 The Fortran flags 351
18.5 Halting 352
18.6 The rounding mode 352
18.7 The underflow mode 353
18.8 The module ieee_exceptions 353
18.8.1 Derived types 353
18.8.2 Inquiring about IEEE exceptions 354
18.8.3 Subroutines for the flags and halting modes 354
18.8.4 Subroutines for the whole of the floating-point status 355
18.9 The module ieee_arithmetic 356
18.9.1 Derived types 356
18.9.2 Inquiring about IEEE arithmetic 356
18.9.3 Elemental functions 358
18.9.4 Non-elemental subroutines 360
18.9.5 Transformational function for kind value 361
18.10 Examples 361
18.10.1 Dot product 361
18.10.2 Calling alternative procedures 362
18.10.3 Calling alternative in-line code 363
18.10.4 Reliable hypotenuse function 363
18.10.5 Access to IEEE arithmetic values 365

19 Interoperability with C **367**
19.1 Introduction 367
19.2 Interoperability of intrinsic types 367
19.3 Interoperability with C pointer types 368
19.4 Interoperability of derived types 370
19.5 Shape and character length disagreement 371
19.6 Interoperability of variables 373
19.7 Function c_sizeof 373
19.8 The value attribute 374
19.9 Interoperability of procedures 375
19.10 Interoperability of global data 377
19.11 Invoking a C function from Fortran 377
19.12 Invoking Fortran from C 378
19.13 Enumerations 381

20 Fortran 2018 coarray enhancements **383**
20.1 Teams 383
20.2 Image failure 384
20.3 Form team statement 385
20.4 Change team construct 385
20.5 Coarrays allocated in teams 386
20.6 Critical construct and image failure 386

20.7 Lock and unlock statements and image failure . . . 386
20.8 Sync team statement . . . 387
20.9 Image selectors . . . 387
20.10 Procedure calls and teams . . . 388
20.11 Intrinsic functions get_team and team_number . . . 388
20.12 Intrinsic function image_index . . . 388
20.13 Intrinsic function num_images . . . 389
20.14 Intrinsic function this_image . . . 390
20.15 Intrinsic function coshape . . . 390
20.16 Intrinsic function move_alloc . . . 390
20.17 Fail image statement . . . 391
20.18 Detecting failed and stopped images . . . 391
20.19 Collective subroutines . . . 392
20.20 New atomic subroutines . . . 393
20.21 Failed images and stat= specifiers . . . 395
20.22 Events . . . 395

21 Fortran 2018 enhancements to interoperability with C **397**
21.1 Introduction . . . 397
21.2 Optional arguments . . . 397
21.3 Low-level C interoperability . . . 399
21.4 Assumed character length . . . 400
21.5 C descriptors . . . 401
21.5.1 Introduction . . . 401
21.5.2 Standard members . . . 401
21.5.3 Argument classification (attribute codes) . . . 402
21.5.4 Argument data type . . . 402
21.5.5 Array layout information . . . 403
21.6 Accessing Fortran objects . . . 404
21.6.1 Traversing contiguous Fortran arrays . . . 404
21.6.2 Generic programming with assumed type . . . 405
21.6.3 Traversing discontiguous Fortran arrays . . . 405
21.6.4 Fortran pointer operations . . . 407
21.6.5 Allocatable objects . . . 409
21.6.6 Handling arrays of any rank . . . 410
21.6.7 Accessing individual array elements via a C descriptor . . . 411
21.6.8 Handling errors from CFI functions . . . 414
21.7 Calling Fortran with C descriptors . . . 414
21.7.1 Allocating storage for a C descriptor . . . 414
21.7.2 Establishing a C descriptor . . . 415
21.7.3 Constructing an array section . . . 416
21.7.4 Accessing components . . . 419
21.8 Restrictions . . . 420
21.8.1 Other limitations on C descriptors . . . 420
21.8.2 Lifetimes of C descriptors . . . 420

21.9 Miscellaneous C interoperability changes . . . 420
21.9.1 Interoperability with the C type ptrdiff_t . . . 420
21.9.2 Relaxation of interoperability requirements . . . 420

22 Fortran 2018 conformance with ISO/IEC/IEEE 60559:2011 423
22.1 Introduction . . . 423
22.2 Subnormal values . . . 423
22.3 Type for floating-point modes . . . 423
22.4 Rounding modes . . . 424
22.5 Rounded conversions . . . 424
22.6 Fused multiply-add . . . 425
22.7 Test sign . . . 425
22.8 Conversion to integer type . . . 425
22.9 Remainder function . . . 425
22.10 Maximum and minimum values . . . 425
22.11 Adjacent machine numbers . . . 426
22.12 Comparisons . . . 426
22.13 Hexadecimal significand input/output . . . 427

23 Minor Fortran 2018 features 429
23.1 Default accessibility for entities accessed from a module . . . 429
23.2 Requiring explicit procedure declarations . . . 429
23.3 Using the properties of an object in its initialization . . . 430
23.4 Generic procedures . . . 430
23.4.1 More concise generic specification . . . 430
23.4.2 Rules for disambiguation . . . 431
23.5 Enhancements to stop and error stop . . . 431
23.6 New intrinsic procedures . . . 431
23.6.1 Checking for unsafe conversions . . . 431
23.6.2 Generalized array reduction . . . 432
23.6.3 Controlling the random number generator . . . 432
23.7 Existing intrinsic procedures . . . 433
23.7.1 Intrinsic function sign . . . 433
23.7.2 Intrinsic functions extends_type_of and same_type_as . . . 433
23.7.3 Simplification of calls of the intrinsic function cmplx . . . 433
23.7.4 Remove many argument dim restrictions . . . 434
23.7.5 Kinds of arguments of intrinsic and IEEE procedures . . . 434
23.7.6 Intrinsic subroutines that access the computing environment . . . 435
23.8 Use of non-standard features . . . 435
23.9 Kind of the do variable in implied-do loops . . . 435
23.10 Improving do concurrent performance . . . 436
23.11 Control of host association . . . 437
23.12 Intent in requirements and the value attribute . . . 438
23.13 Pure procedures . . . 438
23.14 Recursive and non-recursive procedures . . . 438

23.15 Input/output 439
23.15.1 More minimal field width editing 439
23.15.2 Recovering from input format errors 439
23.15.3 Advancing input with size= 439
23.15.4 Precision of stat= variables 439
23.15.5 Connect a file to more than one unit 439
23.15.6 Enhancements to inquire 440
23.15.7 Asynchronous communication 440
23.16 Assumed rank 440
23.16.1 Assumed-rank objects 440
23.16.2 The select rank construct 442

A Deprecated features **445**
A.1 Introduction 445
A.2 Storage association 445
A.3 Alternative form of relational operator 446
A.4 The include line 447
A.5 The do while statement 447
A.6 Double precision real 448
A.7 The dimension, codimension, and parameter statements 448
A.8 Non-default mapping for implicit typing 449
A.9 Fortran 2008 deprecated features 450
A.9.1 The sync memory statement, and atomic_define and atomic_ref . . . 450
A.9.2 Components of type c_ptr or c_funptr 453
A.9.3 Type declarations 453
A.9.4 Denoting absent arguments 454
A.9.5 Alternative form of complex constant 455

B Obsolescent and deleted features **457**
B.1 Features obsolescent in Fortran 95 457
B.1.1 Fixed source form 457
B.1.2 Computed go to 458
B.1.3 Character length specification with character* 458
B.1.4 Data statements among executables 458
B.1.5 Statement functions 459
B.1.6 Assumed character length of function results 460
B.1.7 Alternate return 460
B.2 Feature obsolescent in Fortran 2008: Entry statement 462
B.3 Features obsolescent in Fortran 2018 463
B.3.1 The forall statement and construct 463
B.3.2 The equivalence statement 466
B.3.3 The common block 468
B.3.4 The block data program unit 470
B.3.5 The labelled do construct 471
B.3.6 Specific names of intrinsic procedures 472

B.4 Features deleted in Fortran 95 474
B.5 Feature deleted in Fortran 2003: Carriage control 475
B.6 Features deleted in Fortran 2018 475

C Object-oriented list example **477**

D Solutions to exercises **485**

Index **507**