Modern Fortran Explained

Michael Metcalf
Formerly of CERN, Geneva, Switzerland

John Reid
JKR Associates, Oxfordshire

and

Malcolm Cohen
The Numerical Algorithms Group, Oxfordshire

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 . . . 3
1.4 Language evolution . . . 4
1.5 Fortran 95 . . . 4
1.6 Extensions to Fortran 95 . . . 5
1.7 Fortran 2003 . . . 6
1.8 Fortran 2008 . . . 7
1.9 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 . . . 14
2.6.1 Integer literal constants . . . 14
2.6.2 Real literal constants . . . 15
2.6.3 Complex literal constants . . . 17
2.6.4 Character literal constants . . . 17
2.6.5 Logical literal constants . . . 19
2.7 Names . . . 20
2.8 Scalar variables of intrinsic type . . . 20
2.9 Derived data types . . . 21
2.10 Arrays of intrinsic type . . . 23
2.11 Character substrings . . . 26
2.12 Objects and subobjects . . . 27
2.13 Pointers . . . 27
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 . . . 38
3.5 Scalar relational operators . . . 38
3.6 Scalar logical expressions and assignments . . . 39
3.7 Scalar character expressions and assignments . . . 41
3.8 Structure constructors and scalar defined operators . . . 42
3.9 Scalar defined assignments . . . 45
3.10 Array expressions . . . 46
3.11 Array assignment . . . 48
3.12 Pointers in expressions and assignments . . . 48
3.13 The nullify statement . . . 51
3.14 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 The go to statement . . . 62
4.6 Summary . . . 63

5 Program units and procedures **67**
5.1 Introduction . . . 67
5.2 Main program . . . 68
5.3 The stop statement . . . 69
5.4 External subprograms . . . 69
5.5 Modules . . . 70
5.6 Internal subprograms . . . 73
5.7 Arguments of procedures . . . 73
5.7.1 Pointer arguments . . . 75
5.7.2 Restrictions on actual arguments . . . 76
5.7.3 Arguments with the target attribute . . . 76
5.8 The return statement . . . 77
5.9 Argument intent . . . 77
5.10 Functions . . . 78
5.10.1 Prohibited side-effects . . . 80
5.11 Explicit and implicit interfaces . . . 80
5.12 Procedures as arguments . . . 82
5.13 Keyword and optional arguments . . . 83
5.14 Scope of labels . . . 85
5.15 Scope of names . . . 85
5.16 Direct recursion . . . 88
5.17 Indirect recursion . . . 89
5.18 Overloading and generic interfaces . . . 90

5.19 Assumed character length 93
5.20 The subroutine and function statements 95
5.21 Summary 96

6 Array features **99**
6.1 Introduction 99
6.2 Zero-sized arrays 99
6.3 Assumed-shape arrays 100
6.4 Automatic objects 100
6.5 Allocation of data 102
6.5.1 The allocatable attribute 102
6.5.2 The allocate statement 103
6.5.3 The deallocate statement 104
6.5.4 Allocatable dummy arguments 105
6.5.5 Allocatable functions 105
6.5.6 Allocatable components 106
6.5.7 Allocatable arrays vs. pointers 109
6.6 Elemental operations and assignments 110
6.7 Array-valued functions 110
6.8 The where statement and construct 111
6.9 The forall statement and construct 114
6.10 Pure procedures 117
6.11 Elemental procedures 118
6.12 Array elements 119
6.13 Array subobjects 120
6.14 Arrays of pointers 123
6.15 Pointers as aliases 124
6.16 Array constructors 125
6.17 Mask arrays 126
6.18 Summary 127

7 Specification statements **133**
7.1 Introduction 133
7.2 Implicit typing 133
7.3 Declaring entities of differing shapes 134
7.4 Named constants and constant expressions 134
7.5 Initial values for variables 137
7.5.1 Initialization in type declaration statements 137
7.5.2 The data statement 138
7.5.3 Pointer initialization and the function null 140
7.5.4 Default initialization of components 141
7.6 The public and private attributes 142
7.7 The pointer, target, and allocatable statements 144
7.8 The intent and optional statements 144
7.9 The save attribute 145

7.10 The use statement . . . 146
7.11 Derived-type definitions . . . 149
7.12 The type declaration statement . . . 150
7.13 Type and type parameter specification . . . 152
7.14 Specification expressions . . . 153
7.14.1 Specification functions . . . 153
7.15 The namelist statement . . . 155
7.16 Summary . . . 156

8 Intrinsic procedures **161**
8.1 Introduction . . . 161
8.1.1 Keyword calls . . . 161
8.1.2 Categories of intrinsic procedures . . . 162
8.1.3 The intrinsic statement . . . 162
8.1.4 Argument intents . . . 162
8.2 Inquiry functions for any type . . . 162
8.3 Elemental numeric functions . . . 163
8.3.1 Elemental functions that may convert . . . 163
8.3.2 Elemental functions that do not convert . . . 164
8.4 Elemental mathematical functions . . . 165
8.5 Elemental character and logical functions . . . 166
8.5.1 Character–integer conversions . . . 166
8.5.2 Lexical comparison functions . . . 167
8.5.3 String-handling elemental functions . . . 167
8.5.4 Logical conversion . . . 168
8.6 Non-elemental string-handling functions . . . 168
8.6.1 String-handling inquiry function . . . 168
8.6.2 String-handling transformational functions . . . 168
8.7 Numeric inquiry and manipulation functions . . . 168
8.7.1 Models for integer and real data . . . 168
8.7.2 Numeric inquiry functions . . . 169
8.7.3 Elemental functions to manipulate reals . . . 170
8.7.4 Transformational functions for kind values . . . 171
8.8 Bit manipulation procedures . . . 171
8.8.1 Inquiry function . . . 172
8.8.2 Elemental functions . . . 172
8.8.3 Elemental subroutine . . . 173
8.9 Transfer function . . . 173
8.10 Vector and matrix multiplication functions . . . 174
8.11 Transformational functions that reduce arrays . . . 175
8.11.1 Single argument case . . . 175
8.11.2 Optional argument dim . . . 175
8.11.3 Optional argument mask . . . 176
8.12 Array inquiry functions . . . 176
8.12.1 Allocation status . . . 176

8.12.2 Bounds, shape, and size . . . 176
8.13 Array construction and manipulation functions . . . 177
8.13.1 The merge elemental function . . . 177
8.13.2 Packing and unpacking arrays . . . 177
8.13.3 Reshaping an array . . . 177
8.13.4 Transformational function for replication . . . 178
8.13.5 Array shifting functions . . . 178
8.13.6 Matrix transpose . . . 179
8.14 Transformational functions for geometric location . . . 179
8.15 Transformational function for pointer disassociation . . . 179
8.16 Non-elemental intrinsic subroutines . . . 180
8.16.1 Real-time clock . . . 180
8.16.2 CPU time . . . 181
8.16.3 Random numbers . . . 181
8.17 Summary . . . 182

9 Data transfer **185**
9.1 Introduction . . . 185
9.2 Number conversion . . . 185
9.3 I/O lists . . . 186
9.4 Format definition . . . 188
9.5 Unit numbers . . . 190
9.6 Internal files . . . 191
9.7 Formatted input . . . 193
9.8 Formatted output . . . 194
9.9 List-directed I/O . . . 195
9.10 Namelist I/O . . . 197
9.11 Non-advancing I/O . . . 198
9.12 Edit descriptors . . . 200
9.12.1 Repeat counts . . . 200
9.12.2 Data edit descriptors . . . 201
9.12.3 Character string edit descriptor . . . 205
9.12.4 Control edit descriptors . . . 205
9.13 Unformatted I/O . . . 208
9.14 Direct-access files . . . 209
9.15 Execution of a data transfer statement . . . 210
9.16 Summary . . . 211

10 Operations on external files **213**
10.1 Introduction . . . 213
10.2 Positioning statements for sequential files . . . 214
10.2.1 The backspace statement . . . 214
10.2.2 The rewind statement . . . 214
10.2.3 The endfile statement . . . 215
10.2.4 Data transfer statements . . . 215

10.3 The open statement 216
10.4 The close statement 218
10.5 The inquire statement 219
10.6 Summary 222

11 Floating-point exception handling **223**
11.1 Introduction 223
11.2 The IEEE standard 224
11.3 Access to the features 225
11.4 The Fortran flags 227
11.5 Halting 228
11.6 The rounding mode 228
11.7 The underflow mode (Fortran 2003 only) 229
11.8 The module ieee_exceptions 229
11.8.1 Derived types 229
11.8.2 Inquiry functions for IEEE exceptions 230
11.8.3 Subroutines for the flags and halting modes 230
11.8.4 Subroutines for the whole of the floating-point status 231
11.9 The module ieee_arithmetic 232
11.9.1 Derived types 232
11.9.2 Inquiry functions for IEEE arithmetic 232
11.9.3 Elemental functions 234
11.9.4 Non-elemental subroutines 235
11.9.5 Transformational function for kind value 236
11.10 Examples 237
11.10.1 Dot product 237
11.10.2 Calling alternative procedures 237
11.10.3 Calling alternative in-line code 238
11.10.4 Reliable hypotenuse function 238
11.10.5 Access to IEEE arithmetic values 239

12 Interoperability with C **243**
12.1 Introduction 243
12.2 Interoperability of intrinsic types 243
12.3 Interoperability with C pointer types 245
12.4 Interoperability of derived types 246
12.5 Interoperability of variables 247
12.6 The value attribute 248
12.7 Interoperability of procedures 249
12.8 Interoperability of global data 250
12.9 Invoking a C function from Fortran 251
12.10 Invoking Fortran from C 252
12.11 Enumerations 253

13 Type parameters and procedure pointers **255**
13.1 Introduction . . . 255
13.2 Deferred type parameters . . . 255
13.3 Type parameter enquiry . . . 256
13.4 Parameterized derived types . . . 256
13.4.1 Defining a parameterized derived type . . . 256
13.4.2 Assumed and deferred type parameters . . . 258
13.4.3 Default type parameter values . . . 258
13.4.4 Derived type parameter enquiry . . . 259
13.5 Abstract interfaces . . . 259
13.6 Procedure pointers . . . 261
13.6.1 Procedure pointer variables . . . 261
13.6.2 Procedure pointer components . . . 261
13.6.3 The pass attribute . . . 262

14 Object-oriented programming **265**
14.1 Introduction . . . 265
14.2 Type extension . . . 265
14.2.1 Type extension and type parameters . . . 267
14.3 Polymorphic entities . . . 267
14.3.1 Establishing the dynamic type . . . 268
14.3.2 Limitations on the use of a polymorphic variable . . . 269
14.3.3 Polymorphic arrays and scalars . . . 269
14.3.4 Unlimited polymorphic entities . . . 269
14.3.5 Polymorphic entities and generic resolution . . . 270
14.4 The associate construct . . . 271
14.5 The select type construct . . . 272
14.6 Type-bound procedures . . . 274
14.6.1 Specific type-bound procedures . . . 274
14.6.2 Generic type-bound procedures . . . 277
14.6.3 Type extension and type-bound procedures . . . 279
14.7 Deferred bindings and abstract types . . . 280
14.8 Finalization . . . 281
14.8.1 Type extension and final subroutines . . . 284
14.9 Procedure encapsulation example . . . 284
14.10 Type inquiry functions . . . 286

15 Establishing and moving data **289**
15.1 Introduction . . . 289
15.2 Mixed component accessibility . . . 289
15.3 Structure constructors . . . 289
15.4 The allocate statement . . . 291
15.4.1 Typed allocation and deferred type parameters . . . 291
15.4.2 Polymorphic variables and typed allocation . . . 292
15.4.3 Sourced allocation . . . 292

15.5 Allocatable entities . . . 293
15.5.1 Allocatable scalars . . . 294
15.5.2 Assignment to an allocatable array . . . 294
15.5.3 Transferring an allocation . . . 295
15.6 Pointer assignment . . . 296
15.7 More control of access from a module . . . 296
15.8 Renaming operators on the use statement . . . 297
15.9 Array constructor syntax . . . 297
15.10 Specification and constant expressions . . . 298

16 Miscellaneous enhancements **301**
16.1 Introduction . . . 301
16.2 Pointer intent . . . 301
16.3 The volatile attribute . . . 301
16.3.1 Volatile semantics . . . 302
16.3.2 Volatile scoping . . . 303
16.3.3 Volatile arguments . . . 304
16.4 The import statement . . . 304
16.5 Intrinsic modules . . . 306
16.6 Access to the computing environment . . . 307
16.6.1 Environment variables . . . 307
16.6.2 Information about the program invocation . . . 308
16.7 Support for internationalization . . . 308
16.7.1 Character sets . . . 309
16.7.2 ASCII character set . . . 309
16.7.3 ISO 10646 character set . . . 310
16.7.4 UTF-8 files . . . 310
16.7.5 Decimal comma for input/output . . . 311
16.8 Lengths of names and statements . . . 312
16.9 Binary, octal, and hexadecimal constants . . . 312
16.10 Other changes to intrinsic procedures . . . 313
16.11 Error message retrieval . . . 314
16.12 Enhanced complex constants . . . 314
16.13 Interface block extensions . . . 314
16.14 Public entities of private type . . . 315

17 Input/output enhancements **317**
17.1 Introduction . . . 317
17.2 Non-default derived-type input/output . . . 317
17.3 Asynchronous input/output . . . 320
17.4 The asynchronous attribute . . . 322
17.5 Input and output of IEEE exceptional values . . . 323
17.6 Stream access input/output . . . 323
17.7 Recursive input/output . . . 324
17.8 The flush statement . . . 324

17.9 Comma after a P edit descriptor . . . 324
17.10 The iomsg= specifier . . . 325
17.11 The round= specifier . . . 325
17.12 The sign= specifier . . . 325
17.13 Kind type parameters of integer and logical specifiers . . . 325
17.14 More specifiers in read and write statements . . . 326
17.15 Intrinsic functions for I/O status testing . . . 326
17.16 Some inquire statement enhancements . . . 326
17.17 Namelist enhancements . . . 327

18 Enhanced module facilities **329**
18.1 Introduction . . . 329
18.2 Submodules . . . 330
18.2.1 Separate module procedures . . . 330
18.2.2 Submodules of submodules . . . 331
18.2.3 Submodule entities . . . 331
18.2.4 Submodules and use association . . . 332
18.3 The advantages of submodules . . . 332

19 Coarrays **333**
19.1 Introduction . . . 333
19.2 Referencing images . . . 334
19.3 The properties of coarrays . . . 335
19.4 Accessing coarrays . . . 336
19.5 The sync all statement . . . 337
19.6 Coarrays in procedures . . . 338
19.7 Allocatable coarrays . . . 340
19.8 Coarrays with allocatable or pointer components . . . 341
19.8.1 Data components . . . 341
19.8.2 Procedure pointer components . . . 342
19.9 Coarray components . . . 342
19.10 References to polymorphic subobjects . . . 343
19.11 Volatile and asynchronous attributes . . . 343
19.12 Interoperability . . . 343
19.13 Synchronization . . . 343
19.13.1 Execution segments . . . 343
19.13.2 The sync images statement . . . 344
19.13.3 The lock and unlock statements . . . 345
19.13.4 Critical sections . . . 347
19.13.5 The sync memory statement and atomic subroutines . . . 347
19.13.6 The stat= and errmsg= specifiers in synchronization statements . . . 348
19.13.7 The image control statements . . . 348
19.14 Program termination . . . 348
19.15 Input/output . . . 349
19.16 Intrinsic procedures . . . 351

19.16.1 Inquiry functions . . . 351
19.16.2 Transformational functions . . . 351

20 Other Fortran 2008 enhancements **353**
20.1 Trivial syntactic conveniences . . . 353
20.1.1 Implied-shape arrays . . . 353
20.1.2 Implied-do loops in data statements . . . 353
20.1.3 Type-bound procedures . . . 354
20.1.4 Structure constructors . . . 354
20.1.5 Semicolons . . . 355
20.1.6 The stop statement . . . 355
20.1.7 Exit from nearly any construct . . . 355
20.2 Limitation changes . . . 356
20.2.1 64-bit integer support . . . 356
20.2.2 Maximum array rank . . . 356
20.3 Data expressiveness . . . 356
20.3.1 Allocatable components of recursive type . . . 356
20.3.2 Initial pointer association . . . 358
20.4 Performance-oriented features . . . 359
20.4.1 The do concurrent construct . . . 359
20.4.2 The contiguous attribute . . . 361
20.4.3 Simply contiguous array designators . . . 364
20.5 Computational expressiveness . . . 365
20.5.1 Accessing parts of complex variables . . . 365
20.5.2 Pointer functions denoting variables . . . 366
20.5.3 The block construct . . . 366
20.5.4 Impure elemental procedures . . . 368
20.5.5 Internal procedures as actual arguments . . . 370
20.5.6 Specifying the kind of a forall index variable . . . 370
20.5.7 Generic resolution . . . 371
20.6 Data usage and computation . . . 372
20.6.1 Enhancements to the allocate statement . . . 372
20.6.2 Automatic reallocation . . . 373
20.6.3 Elemental subprogram restrictions . . . 373
20.7 Input/output . . . 374
20.7.1 Recursive input/output . . . 374
20.7.2 The newunit= specifier . . . 374
20.7.3 Writing comma-separated values . . . 375
20.8 Intrinsic procedures . . . 376
20.9 Mathemetical intrinsic functions . . . 376
20.9.1 Changes to trigonometric functions . . . 376
20.9.2 New hyperbolic trigonometic functions . . . 376
20.9.3 New special mathematical functions . . . 377
20.9.4 Euclidean norms . . . 378
20.10 Bit manipulation . . . 378

20.10.1 Bitwise (unsigned) comparison . . . 378
20.10.2 Double-width shifting . . . 379
20.10.3 Bitwise reductions . . . 379
20.10.4 Counting bits . . . 380
20.10.5 Producing bitmasks . . . 380
20.10.6 Merging bits . . . 381
20.10.7 Additional shift operations . . . 381
20.11 Miscellaneous intrinsic procedures . . . 382
20.11.1 Procedures supporting coarrays . . . 382
20.11.2 Executing another program . . . 382
20.11.3 Character comparison . . . 383
20.11.4 Array searching . . . 383
20.11.5 Logical parity . . . 383
20.11.6 Decimal arithmetic support . . . 384
20.11.7 Size of an object in memory . . . 384
20.12 Additions to the iso_fortran_env module . . . 385
20.12.1 Compilation information . . . 385
20.12.2 Names for common kinds . . . 385
20.12.3 Kind arrays . . . 386
20.12.4 Coarray support facilities . . . 386
20.13 Changes to other standard intrinsic modules . . . 387
20.13.1 The iso_c_binding module . . . 387
20.13.2 The ieee_arithmetic module . . . 387
20.14 Programs and procedures . . . 388
20.14.1 Saved module entities . . . 388
20.14.2 Automatic pointer targetting . . . 388
20.14.3 Denoting absent arguments . . . 389

A Intrinsic procedures **393**

B Deprecated features **399**
B.1 Introduction . . . 399
B.2 Storage association . . . 399
B.2.1 Storage units . . . 399
B.2.2 The equivalence statement . . . 400
B.2.3 The common block . . . 402
B.2.4 The block data program unit . . . 404
B.2.5 Coarrays and storage association . . . 405
B.3 Shape and character length disagreement . . . 405
B.4 The include line . . . 407
B.5 Other forms of loop control . . . 407
B.5.1 The labelled do construct . . . 407
B.5.2 The do while . . . 408
B.6 Double precision real . . . 408
B.7 The dimension, codimension, and parameter statements . . . 409

B.8 Specific names of intrinsic procedures . . . 410
B.9 Non-default mapping for implicit typing . . . 412
B.10 Fortran 2008 deprecated features . . . 413
B.10.1 The sync memory statement and atomic subroutines . . . 413
B.10.2 Components of type `c_ptr` or `c_funptr` . . . 416
B.10.3 Type declarations . . . 416
B.10.4 Redundant contains statement . . . 417
B.10.5 The end statement . . . 417
B.10.6 Referencing `atan2` by the name `atan` . . . 418

C Obsolescent features **419**
C.1 Obsolescent in Fortran 95 . . . 419
C.1.1 Fixed source form . . . 419
C.1.2 Computed go to . . . 420
C.1.3 Character length specification character* . . . 420
C.1.4 Data statements among executables . . . 420
C.1.5 Statement functions . . . 421
C.1.6 Assumed character length of function results . . . 422
C.1.7 Arithmetic if statement . . . 422
C.1.8 Shared do-loop termination . . . 423
C.1.9 Alternate return . . . 423
C.2 Feature obsolescent in Fortran 2008: Entry statement . . . 424
C.3 Feature deleted in Fortran 2003: Carriage control . . . 426
C.4 Features deleted in Fortran 95 . . . 427

D Avoiding compilation cascades **429**

E Object-oriented list example **433**

F Fortran terms **441**

G Solutions to exercises **453**

Index **475**