Fortran 95/2003 Explained

Michael Metcalf
Formerly of CERN, Geneva, Switzerland

John Reid
JKR Associates, Oxfordshire

and

Malcolm Cohen
The Numerical Algorithms Group
Oxfordshire

Scanned by terminator

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

Great Clarendon Street, Oxford OX2 6DP

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide in

Oxford New York

Auckland Cape Town Dar es Salaam Hong Kong Karachi
Kuala Lumpur Madrid Melbourne Mexico City Nairobi
New Delhi Shanghai Taipei Toronto

With offices in

Argentina Austria Brazil Chile Czech Republic France Greece
Guatemala Hungary Italy Japan South Korea Poland Portugal
Singapore Switzerland Thailand Turkey Ukraine Vietnam

Oxford is a registered trade mark of Oxford University Press
in the UK and in certain other countries

Published in the United States
by Oxford University Press Inc., New York

© Oxford University Press 2004

The moral rights of the author have been asserted

Database right Oxford University Press (maker)

First published 2004
Reprinted 2004 (with corrections), 2005 (with corrections)

All rights reserved. No part of this publication may be reproduced,
stored in a retrieval system, or transmitted, in any form or by any means,
without the prior permission in writing of Oxford University Press,
or as expressly permitted by law, or under terms agreed with the appropriate
reprographics rights organization. Enquiries concerning reproduction
outside the scope of the above should be sent to the Rights Department,
Oxford University Press, at the address above.

You must not circulate this book in any other binding or cover
and you must impose this same condition on any acquirer.

A catalogue record for this title is available from the British Library

Library of Congress Cataloging in Publication Data
(Data available)

ISBN 0 19 852692 X (Hbk)
ISBN 0 19 852693 8 (Pbk)

10 9 8 7 6 5 4 3

Printed in Great Britain
on acid-free paper by
Biddles Ltd., King's Lynn, Norfolk

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 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 17

2.6.5 Logical literal constants 19

2.7 Names 19

2.8 Scalar variables of intrinsic type 20

2.9 Derived data types 21

2.10 Arrays of intrinsic type 22

2.11 Character substrings 25

2.12 Objects and subobjects 26

2.13 Pointers 27

2.14 Summary 28

2.15 Exercises 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 Summary 51
3.14 Exercises 51

4 Control constructs 53
4.1 Introduction 53
4.2 The go to statement 53
4.3 The if statement and construct 54
4.4 The case construct 56
4.5 The do construct 58
4.6 Summary 63
4.7 Exercises 65

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 77
5.8 The return statement 77
5.9 Argument intent 77
5.10 Functions 79
5.10.1 Prohibited side-effects 80
5.11 Explicit and implicit interfaces 81
5.12 Procedures as arguments 83
5.13 Keyword and optional arguments 84
5.14 Scope of labels 85
5.15 Scope of names 86
5.16 Direct recursion 88
5.17 Indirect recursion 89

5.18 Overloading and generic interfaces 91
5.19 Assumed character length 95
5.20 The subroutine and function statements 96
5.21 Summary 96
5.22 Exercises 97

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 Heap storage 102
6.5.1 Allocatable arrays 102
6.5.2 The allocate statement 103
6.5.3 The deallocate statement 105
6.5.4 The nullify statement 105
6.6 Elemental operations and assignments 106
6.7 Array-valued functions 106
6.8 The where statement and construct 107
6.9 The forall statement and construct 110
6.10 Pure procedures 113
6.11 Elemental procedures 114
6.12 Array elements 116
6.13 Array subobjects 117
6.14 Arrays of pointers 120
6.15 Pointers as aliases 121
6.16 Array constructors 122
6.17 Mask arrays 123
6.18 Summary 124
6.19 Exercises 127

7 Specification statements **129**
7.1 Introduction 129
7.2 Implicit typing 129
7.3 Declaring entities of differing shapes 132
7.4 Named constants and initialization expressions 132
7.5 Initial values for variables 135
7.5.1 Initialization in type declaration statements 135
7.5.2 The data statement 135
7.5.3 Pointer initialization and the function null 138
7.5.4 Default initialization of components 139
7.6 The public and private attributes 140
7.7 The pointer, target, and allocatable statements 142
7.8 The intent and optional statements 142
7.9 The save attribute 143

7.10 The use statement 144
7.11 Derived-type definitions 147
7.12 The type declaration statement 149
7.13 Type and type parameter specification 150
7.14 Specification expressions 151
7.14.1 Specification functions 152
7.15 The namelist statement 153
7.16 Summary 154
7.17 Exercises 156

8 Intrinsic procedures 159
8.1 Introduction 159
8.1.1 Keyword calls 159
8.1.2 Categories of intrinsic procedures 160
8.1.3 The intrinsic statement 160
8.1.4 Argument intents 160
8.2 Inquiry functions for any type 160
8.3 Elemental numeric functions 161
8.3.1 Elemental functions that may convert 161
8.3.2 Elemental functions that do not convert 162
8.4 Elemental mathematical functions 163
8.5 Elemental character and logical functions 164
8.5.1 Character–integer conversions 164
8.5.2 Lexical comparison functions 164
8.5.3 String-handling elemental functions 165
8.5.4 Logical conversion 165
8.6 Non-elemental string-handling functions 166
8.6.1 String-handling inquiry function 166
8.6.2 String-handling transformational functions 166
8.7 Numeric inquiry and manipulation functions 166
8.7.1 Models for integer and real data 166
8.7.2 Numeric inquiry functions 167
8.7.3 Elemental functions to manipulate reals 168
8.7.4 Transformational functions for kind values 169
8.8 Bit manipulation procedures 169
8.8.1 Inquiry function 169
8.8.2 Elemental functions 170
8.8.3 Elemental subroutine 171
8.9 Transfer function 171
8.10 Vector and matrix multiplication functions 172
8.11 Transformational functions that reduce arrays 172
8.11.1 Single argument case 172
8.11.2 Optional argument dim 173
8.11.3 Optional argument mask 173
8.12 Array inquiry functions 174

8.12.1 Allocation status 174
8.12.2 Bounds, shape, and size 174
8.13 Array construction and manipulation functions 174
8.13.1 The merge elemental function 174
8.13.2 Packing and unpacking arrays 175
8.13.3 Reshaping an array 175
8.13.4 Transformational function for replication 176
8.13.5 Array shifting functions 176
8.13.6 Matrix transpose 176
8.14 Transformational functions for geometric location 176
8.15 Transformational function for pointer disassociation 177
8.16 Non-elemental intrinsic subroutines 177
8.16.1 Real-time clock 177
8.16.2 CPU time 178
8.16.3 Random numbers 179
8.17 Summary 179
8.18 Exercises 180

9 Data transfer **181**
9.1 Introduction 181
9.2 Number conversion 181
9.3 I/O lists 182
9.4 Format definition 184
9.5 Unit numbers 186
9.6 Internal files 187
9.7 Formatted input 188
9.8 Formatted output 190
9.9 List-directed I/O 190
9.10 Namelist I/O 192
9.11 Non-advancing I/O 194
9.12 Edit descriptors 195
9.12.1 Repeat counts 195
9.12.2 Data edit descriptors 197
9.12.3 Minimal field width editing 199
9.12.4 Character string edit descriptor 200
9.12.5 Control edit descriptors 200
9.13 Unformatted I/O 203
9.14 Direct-access files 203
9.15 Execution of a data transfer statement 205
9.16 Summary 206
9.17 Exercises 207

10 Operations on external files **209**
10.1 Introduction 209
10.2 File positioning statements 210

10.2.1 The backspace statement . . . 210
10.2.2 The rewind statement . . . 210
10.2.3 The endfile statement . . . 211
10.2.4 Data transfer statements . . . 211
10.3 The open statement . . . 212
10.4 The close statement . . . 214
10.5 The inquire statement . . . 215
10.6 Summary . . . 218
10.7 Exercises . . . 218

11 Floating-point exception handling 219
11.1 Introduction . . . 219
11.2 The IEEE standard . . . 220
11.3 Access to the features . . . 221
11.4 The Fortran flags . . . 223
11.5 Halting . . . 224
11.6 The rounding mode . . . 224
11.7 The underflow mode (Fortran 2003 only) . . . 225
11.8 The module ieee_exceptions . . . 225
11.8.1 Derived types . . . 225
11.8.2 Inquiry functions for IEEE exceptions . . . 226
11.8.3 Subroutines for the flags and halting modes . . . 226
11.8.4 Subroutines for the whole of the floating-point status . . . 227
11.9 The module ieee_arithmetic . . . 228
11.9.1 Derived types . . . 228
11.9.2 Inquiry functions for IEEE arithmetic . . . 229
11.9.3 Elemental functions . . . 230
11.9.4 Non-elemental subroutines . . . 232
11.9.5 Transformational function for kind value . . . 232
11.10 Examples . . . 233
11.10.1 Dot product . . . 233
11.10.2 Calling alternative procedures . . . 234
11.10.3 Calling alternative in-line code . . . 234
11.10.4 Reliable hypotenuse function . . . 235
11.10.5 Access to IEEE arithmetic values . . . 237

12 Allocatable array extensions 239
12.1 Introduction . . . 239
12.2 Allocatable dummy arguments . . . 240
12.3 Allocatable functions . . . 240
12.4 Allocatable components . . . 241
12.5 Exercises . . . 244

13 Enhanced module facilities 245
13.1 Introduction . . . 245

13.2 Submodules 246
13.2.1 Separate module procedures 246
13.2.2 Submodules of submodules 247
13.2.3 Submodule entities 247
13.2.4 Submodules and use association 248
13.3 The advantages of submodules 248

14 Interoperability with C 249
14.1 Introduction 249
14.2 Interoperability of intrinsic types 249
14.3 Interoperability with C pointer types 251
14.4 Interoperability of derived types 252
14.5 Interoperability of variables 253
14.6 The value attribute 254
14.7 Interoperability of procedures 255
14.8 Interoperability of global data 256
14.9 Invoking a C function from Fortran 257
14.10 Invoking Fortran from C 257
14.11 Enumerations 260

15 Type parameters and procedure pointers 261
15.1 Introduction 261
15.2 Deferred type parameters 261
15.3 Type parameter enquiry 262
15.4 Parameterized derived types 262
15.4.1 Defining a parameterized derived type 262
15.4.2 Assumed and deferred type parameters 264
15.4.3 Default type parameter values 264
15.4.4 Derived type parameter enquiry 265
15.5 Abstract interfaces 265
15.6 Procedure pointers 267
15.6.1 Procedure pointer variables 267
15.6.2 Procedure pointer components 267
15.6.3 The pass attribute 268
15.7 Exercises 270

16 Object-oriented programming 271
16.1 Introduction 271
16.2 Type extension 271
16.2.1 Type extension and type parameters 273
16.3 Polymorphic entities 273
16.3.1 Unlimited polymorphic entities 274
16.4 The associate construct 275
16.5 The select type construct 276
16.6 Type-bound procedures 279

16.6.1 Specific type-bound procedures . . . 279
16.6.2 Generic type-bound procedures . . . 281
16.6.3 Type extension and type-bound procedures . . . 282
16.7 Deferred bindings and abstract types . . . 284
16.8 Finalization . . . 284
16.8.1 Type extension and final subroutines . . . 286
16.9 Procedure encapsulation example . . . 288
16.10 Type inquiry functions . . . 290
16.11 Exercises . . . 291

17 Establishing and moving data **293**
17.1 Introduction . . . 293
17.2 Mixed component accessibility . . . 293
17.3 Structure constructors . . . 293
17.4 The allocate statement . . . 295
17.4.1 Typed allocation and deferred type parameters . . . 295
17.4.2 Polymorphic variables and typed allocation . . . 296
17.4.3 Sourced allocation . . . 297
17.5 Allocatable entities . . . 297
17.5.1 Allocatable scalars . . . 298
17.5.2 Assignment to an allocatable array . . . 298
17.5.3 Transferring an allocation . . . 299
17.6 Pointer assignment . . . 300
17.7 More control of access from a module . . . 300
17.8 Renaming operators on the use statement . . . 301
17.9 Array constructor syntax . . . 302
17.10 Specification and initialization expressions . . . 303
17.11 Exercise . . . 303

18 Miscellaneous enhancements **305**
18.1 Introduction . . . 305
18.2 Pointer intent . . . 305
18.3 The volatile attribute . . . 305
18.3.1 Volatile semantics . . . 306
18.3.2 Volatile scoping . . . 307
18.3.3 Volatile arguments . . . 308
18.4 The import statement . . . 308
18.5 Intrinsic modules . . . 310
18.6 Access to the computing environment . . . 311
18.6.1 Environment variables . . . 311
18.6.2 Command arguments and the command line . . . 312
18.7 Support for internationalization . . . 312
18.7.1 Character sets . . . 313
18.7.2 ASCII character set . . . 313
18.7.3 ISO 10646 character set . . . 314

18.7.4 UTF-8 files 314
18.7.5 Decimal comma for input/output 315
18.8 Lengths of names and statements 316
18.9 Binary, octal, and hexadecimal constants 316
18.10 Other changes to intrinsic functions 317
18.11 Error message retrieval 318
18.12 Enhanced complex constants 318
18.13 Interface block extensions 318
18.14 Public entities of private type 319

19 Input/output enhancements 321
19.1 Introduction 321
19.2 Non-default derived-type input/output 321
19.3 Asynchronous input/output 324
19.4 The asynchronous attribute 326
19.5 Input and output of IEEE exceptional values 327
19.6 Stream access input/output 327
19.7 Recursive input/output 328
19.8 The flush statement 328
19.9 Comma after a P edit descriptor 328
19.10 The iomsg= specifier 329
19.11 The round= specifier 329
19.12 The sign= specifier 329
19.13 Kind type parameters of integer specifiers 329
19.14 Intrinsic functions for I/O status testing 329
19.15 Some inquire statement enhancements 330

20 Other features 331
20.1 Introduction 331
20.2 Storage association 331
20.2.1 Storage units 331
20.2.2 The equivalence statement 332
20.2.3 The common block 334
20.2.4 The block data program unit 336
20.3 Shape and character length disagreement 337
20.4 The entry statement 339
20.5 The include line 340
20.6 The do while form of loop control 341
20.7 Double precision real 341
20.8 The dimension and parameter statements 342
20.9 Specific names of intrinsic procedures 343

A Intrinsic procedures 347

B Obsolescent features 353

B.1 Obsolescent in Fortran 95 353
B.1.1 Fixed source form 353
B.1.2 Computed go to 354
B.1.3 Character length specification character* 354
B.1.4 Data statements among executables 355
B.1.5 Statement functions 355
B.1.6 Assumed character length of function results 356
B.1.7 Arithmetic if statement 357
B.1.8 Shared do-loop termination 357
B.1.9 Alternate return 358
B.2 Feature deleted in Fortran 2003: Carriage control 359
B.3 Features deleted in Fortran 95 359

C Pointer example **361**

D Avoiding compilation cascades **371**

E Fortran terms **375**

F Solutions to exercises **391**

Index **405**