Annex D

(informative)

Index

The this index, entries in italics denote BNF terms, entries in bold face denote language keywords, and page numbers in bold denote primary or defining text.

Symbols

! 26
- 87, **101**
% 75
& 26
& **179**
***** 87, **101**
***** (symbol) 47, 50, 56, 62, 138, **145**, **175**, **178**, 198, 208
****** 86, **101**
+ 87, **101**
. 86, 88
.AND. 88, **104**
.EQ. 87, **103**
.EQV. 88, **104**
.FALSE. **37**
.GE. 87, **103**
.GT. 87, **103**
.LE. 87, **103**
.LT. 87, **103**
.NE. 87, **103**
.NEQV. 88, **104**
.NOT. 88, **104**
.OR. 88, **104**
.TRUE. **37**
/ 87, **101**, 163
/ (symbol) 59, 61, 65, 68
// 36, 87, **102**
/= 87, **103**
: 54, 74, 76
:: 47
; 26
< 87, **103**
<= 87, **103**
= 107, 194, 198, 211
== 87, **103**
=> (pointer assignment) 47, 110
=> (rename) 187
> 87, **103**
>= 88, **103**

A

ACCESS= specifier **141**, **157**
accessibility attribute **52**
accessibility statements **58**
access-spec **52**
access-stmt **58**
ACTION= specifier **142**, **159**
actual-arg **198**
ADVANCE= specifier **147**
ALLOCATABLE 47, 60
allocatable array **55**
ALLOCATABLE attribute **57**
ALLOCATABLE statement **60**
allocatable-stmt **60**
ALLOCATE 79
ALLOCATE statement **79**
allocate-stmt **79**
alt-return-spec **198**
APOSTROPHE (DELIM value) 183
argument association **199**, 282
argument keyword **199**
argument keywords 18, 217, 280
arithmetic IF statement **131**
arithmetic-if-stmt **131**
array **17**, **54–56**, **75–78**
    assumed-shape **55**
    assumed-size **56**
    automatic **54**
    explicit-shape **54**
array constructor **45**
array element **17**, **77**
array element order **77**
array pointer **55**

- array section **17**, **77**
- *array-constructor* **45**
- *array-element* **76**
- *array-section* **76**
- *array-spec* **54**
- ASCII collating sequence **37**
- **ASSIGNMENT** 194
- assignment **107–119**
  - defined **108**
  - elemental array (FORALL) 114
  - intrinsic **107**
  - masked array (WHERE) **111**
  - pointer **110**
- assignment statement **107**
- *assignment-stmt* **107**
- association 18
  - argument **199**, 282
  - host **282**
  - name **282**
  - pointer **284**
  - sequence **202**
  - storage **285**
  - use **282**
- association status
  - pointer **284**
- assumed character length parameter **51**
- assumed-shape array **55**
- assumed-size array **56**
- attribute specification statements **57–71**
- attributes **52–57**
  - accessibility **52**
  - ALLOCATABLE **57**
  - DIMENSION **54**
  - EXTERNAL **57**
  - INTENT **53**
  - INTRINSIC **57**
  - OPTIONAL **57**
  - PARAMETER **52**
  - POINTER **57**
  - PRIVATE **52**
  - PUBLIC **52**
  - SAVE **56**, 61
  - TARGET **57**
- *attr-spec* **47**
- automatic array **54**
- automatic data object **49**

B

- **BACKSPACE** 154
- BACKSPACE statement **154**
- *backspace-stmt* **154**
- binary constant - *see* constant, boz
- bit model **219**
- blank common **68**
- **BLANK=** specifier **142**, **158**
- *blank-interp-edit-desc* **163**
- block **121**
- *block* **121**
- **BLOCK DATA** 189
- block data program **189**
- *block-data* **189**
- *block-data-stmt* **189**
- *block-do-construct* **126**
- *boz-literal-constant* **32**
- branch target statement **130**

C

- CALL statement **198**
- *call-stmt* **198**
- **CASE** 123
- CASE construct **123**
- *case-construct* **123**
- *case-stmt* **123**
- CHAR intrinsic 36
- **CHARACTER** **35**, 47
- *character* **21**
- character context 25
- character intrinsic operation **102**
- character literal constant **35**
- character sequence type **39**
- character set **21**
- character string **35**
- character type **35–37**
- CHARACTER type specifier **50**
- characteristics of a procedure **192**
- *char-constant* **23**
- *char-expr* **90**
- *char-len-param-value* **50**
- *char-literal-constant* **35**
- *char-string-edit-desc* **163**
- **CLOSE** 143
- CLOSE statement **143**
- *close-stmt* **143**
- collating sequence **36**
- comment **26**, **27**
- **COMMON** **68**
- common association **70**
- common block **68**, 276, 329
- common block storage sequence 69
- COMMON statement **68–71**
- *common-block-name* **68**
- *common-stmt* **68**

compatibility
    FORTRAN 77 **3**
        3
COMPLEX 34, 47
complex type **34**
COMPLEX type specifier **50**
complex-literal-constant **34**
component-def-stmt **38**
components 280
computed GO TO statement **131**
computed-goto-stmt **131**
concatenation 36
conform 107
conformable **17**
conformance 92
connected files **138**
constant **16**, 23, 30
    binary - *see* constant, boz
    boz **32**
    character **35**
    hexadecimal - *see* constant, boz
    integer **31**
    logical **37**
    named **60**
    octal - *see* constant, boz
constant **23**
constant expression **93**
constant subobject **16**
constructor
    array **45**
    derived-type **44**
    structure **44**
CONTAINS 186, 211
CONTAINS statement **211**
contains-stmt **211**
continuation **26**, **27**
CONTINUE 131
CONTINUE statement **131**
continue-stmt **131**
control edit descriptors **171**
control-edit-desc **163**
conversion
    numeric **108**
current record **136**
CYCLE 129
CYCLE statement 126, **129**
cycle-stmt **129**

D

DATA 61
data edit descriptors **165–171**
data object **16**
data object reference 18
DATA statement **61**, 288
data transfer **151**
data transfer statements **144**
data type 15, **29**–46
    see also type
data type of a primary **91**
data type of an expression **90**
data type of an operation **92**
data type
    concept **29**
data-edit-desc **162**
data-implied-do 61
data-ref **75**
data-stmt **61**
DEALLOCATE 82
DEALLOCATE statement **82**
deallocate-stmt **82**
declaration 18
declarations **47–71**
DEFAULT 124
default character **35**
default complex **34**
default initialization **38**, 288
default integer **31**
default logical **37**
default real **33**
default-char-exp **90**
deferred-shape array **55**
defined 18
defined assignment **196**
defined assignment statement **108**
defined operation **90**, **104**, **196**
defined-binary-op **88**
defined-operator **24**
defined-unary-op **86**
definition **18**
definition of variables **288**
DELIM= specifier **142**, **159**, 182
derived type determination **43**
derived type type specifier **52**
derived types 15, **37–45**
derived-type-def **38**
digit-string **31**
DIMENSION 47, 59
DIMENSION attribute **54**
DIMENSION statement **59**
dimension-stmt **59**
direct access **135**
direct access input/output statement **146**
DIRECT= specifier **158**
disassociated **17**
DO 126

DO construct **126**
DO statement **126**
DO WHILE statement **126**
do-construct **126**
do-stmt **126**
`DOUBLE PRECISION` 33, 47
double precision real **33**
DOUBLE PRECISION type specifier **50**
dummy arguments
  restrictions **203**
dummy procedure **191**
dummy-arg **208**

E

edit descriptors *see* format descriptors
element array assignment (FORALL) 114
`ELEMENTAL` 207
elemental intrinsic procedure **217**
elemental procedure **213**
`ELSE` 122
else-if-stmt **122**
else-stmt **122**
`ELSEWHERE` 112
elsewhere-stmt 112
`END` 185
END statement **14**
`END=` specifier **147**
`ENDFILE` 154
endfile record **134**
ENDFILE statement 134, **155**
endfile-stmt **154**
end-of-file condition **149**
end-of-record condition **149**
end-program-stmt **185**
entity-decl **47**
`ENTRY` 209
entry-stmt **209**
`EOR=` specifier **147**
`EQUIVALENCE` 66
EQUIVALENCE statement **66–68**
equivalence-stmt **66**
`ERR=` specifier **143**, **147**
evaluation
  operations 96
  optional 98
  parentheses 98
executable constructs 121
executable-construct **10**
execution control **121–131**
`EXIST=` specifier **157**
`EXIT` 129
EXIT statement **129**
exit-stmt **129**
explicit formatting **161–174**
explicit interface **193**
explicit-shape array **54**
explicit-shape-spec **54**
expr **88**
expressions **16, 85–106**
extent **17**
`EXTERNAL` 47, 197
EXTERNAL attribute **57**
external file **134**
external procedure 12, **191**
EXTERNAL statement **197**
external subprogram 11
external-stmt **197**
external-subprogram **9**

F

file access **135**
file connection **138**
file inquiry **155**
file position **136**
file positioning statements **154**
`FILE=` specifier **141**, **157**
files
  connected **138**
  external **134**
  internal **137**
  preconnected **139**
fixed source form **27**
`FORALL` 114
FORALL construct **114**
forall-construc **114**
`FORM=` specifier **141**, **158**
`FORMAT` 161
format **145**
format descriptors
  / **173**
  : **173**
  A **170**
  B **166**
  BN **174**
  BZ **174**
  control edit descriptors **171**
  D **167**
  data edit descriptors **165–171**
  E **167**
  EN **168**
  ES **169**
  F **167**

  - G 170, **170**
  - I **166**
  - L **170**
  - O **166**
  - P **173**
  - S **173**
  - scale factor **173**
  - SP **173**
  - SS **173**
  - TL **172**
  - TR **172**
  - X **172**
  - Z **166**
- format specifier **145**
- FORMAT statement 145, **161**
- *format-item* **162**
- *format-specification* **161**
- *format-stmt* **161**
- formatted data transfer **152**
- formatted input/output statement **145**
- formatted record **133**
- **`FORMATTED=`** specifier **158**
- formatting
  - explicit **161–174**
  - list-directed **153**, **174–178**
  - namelist **153**, **178–183**
- FORTRAN 77 compatibility **3**
- Fortran 90 compatibility **3**
- free source form **25**
- **`FUNCTION`** 206
- function 12
- function reference **16**, **205**
- FUNCTION statement **206**
- *function-reference* **198**
- *function-stmt* **206**
- *function-subprogram* **9**, **206**

G

- generic identifier **195**
- generic interface **195**
- generic name **195**
- generic procedure references **277**
- *generic-spec* **194**
- global entities **275**
- **`GO TO`** 131
- GO TO statement **131**
- *goto-stmt* **131**

H

- hexadecimal constant - *see* constant, boz
- host 12, **186**
- host association **282**
- host scoping unit 11

I

- ICHAR intrinsic 36
- **`IF`** 122, 123, 131
- IF construct **122**
- IF statement **123**
- *if-construct* **122**
- *if-stmt* **123**
- *if-then-stmt* **122**
- imaginary part **34**
- **`IMPLICIT`** 63
- implicit interface **198**
- **`IMPLICIT NONE`** 63
- IMPLICIT statement **63**
- *implicit-stmt* **63**
- implied-DO **45**, **61**, **148**, 151
- **`IN`** 53
- **`INCLUDE`** 27
- INCLUDE line **27**
- initial point **136**
- initialization 40, 48, **49**, 288
- *initialization* **47**
- initialization expression **94**
- *initialization-expr* **94**
- **`INOUT`** 53
- input/output editing **161–183**
- input/output list **148**
- input/output statements **133–160**
- *input-item* **148**
- **`INQUIRE`** 155
- INQUIRE statement **155**
- *inquire-stmt* **155**
- inquiry function (intrinsic) **217**
- *int-constant* **23**
- **`INTEGER`** 31, 47
- integer constant **31**
- integer editing **166**
- integer model **219**
- integer type **31–32**
- INTEGER type specifier **50**
- **`INTENT`** 47, 58, 218
- INTENT attribute 38, **53**, 206
- INTENT statement **58**
- *intent-spec* **53**
- *intent-stmt* **58**
- **`INTERFACE`** 193

- interface
  - (procedure) **192**
  - explicit **193**
  - generic **195**
  - implicit **198**
- interface body **194**
- *interface-block* **193**
- *interface-body* **193**
- internal files **137**
- internal procedure 12, **191**
- internal subprogram 11
- *internal-subprogram* **10**
- *int-expr* **90**
- *int-literal-constant* **31**
- **INTRINSIC** 47, 197
- intrinsic 19
  - elemental **217**
  - function **217**
  - inquiry function **217**
  - subroutine **222**
  - transformational **217**
- intrinsic assignment statement **107**
- INTRINSIC attribute **57**
- intrinsic data types **31–37**
- intrinsic operation **89**
- intrinsic operations **101–104**
  - logical 37
- intrinsic procedures **228–274**
  - *see alphabetical listing, ch. 13*
- INTRINSIC statement **197**
- intrinsic type 15
- *intrinsic-operator* **23**
- *intrinsic-stmt* **197**
- IOSTAT= 150
- **IOSTAT=** specifier **143**, **146**
- *io-unit* **138**

K

- keyword 18, **199**
- *keyword* **198**
- **KIND** 47, 50
- KIND intrinsic 31, 32, 34, 35, 37, 218
- kind type parameter 31, 32, 34, 35, 37
- *kind-param* **31**

L

- label 281
- *label* **24**
- **LEN** 50
- length **35**
- line 24
- list-directed formatting **153, 174–178**
- list-directed input/output statement **146**
- literal constant **16**, **73**
- *literal-constant* **23**
- local entities **275**
- **LOGICAL** **37**, 47
- logical constant **37**
- logical intrinsic operations 37, **104**
- logical type **37**
- LOGICAL type specifier **51**
- *logical-exp* **90**
- *logical-literal-constant* **37**
- loop **126**

M

- main program 12, **185**
- *main-program* **9**, **185**
- many-one array section **78**
- masked array assignment (WHERE) **111**
- model
  - bit **219**
  - integer **219**
  - real **219**
- **MODULE** 186
- module 13, **186**
- *module* **9**
- **MODULE PROCEDURE** 194
- module procedure 12
- module reference 18, **187**
- module subprogram 11
- *module-procedure-stmt* **194**
- *module-subprogram* **10**

N

- name 18
- *name* **22**
- name association **282**
- **NAME=** specifier **157**
- named common block **68**
- named constant **16**, **52**, **60**, **73**
- **NAMED=** specifier **157**
- *named-constant* **23**
- **NAMELIST** 65
- namelist formatting **153**, **178–183**
- namelist input/output statement **146**
- NAMELIST statement **65**
- *namelist-stmt* **65**

`NEXTREC=` specifier **158**
`NML=` specifier **146**
nonblock-do-construct **127**
`NONE` *see* **`IMPLICIT NONE`**
`NONE` (DELIM value) 182
nonnumeric types **35**–46
NULL intrinsic 39, 40, 47, 111
`NULLIFY` 82
NULLIFY statement **82**
nullify-stmt **82**
`NUMBER=` specifier **157**
numeric conversion **108**
numeric editing **165**
numeric intrinsic operations **101**
numeric sequence type **39**
numeric storage unit **285**
numeric types **31–35**
numeric-expr **91**

O

object -- *see* data object
octal constant - *see* constant, boz
`ONLY` 187
`OPEN` 140
OPEN statement 139
`OPENED=` specifier **157**
open-stmt **140**
operations 30
  character intrinsic **102**
  defined **104**
  logical intrinsic **104**
  numeric intrinsic **101**
  relational intrinsic **102**
`OPERATOR` 194
operator precedence **105**
operators 23
`OPTIONAL` 47, 58
OPTIONAL attribute **57**
optional dummy argument 203, 218
OPTIONAL statement **58**
optional-stmt **58**
`OUT` 53
output-item **148**

P

`PAD=` specifier **142**, **159**
PARAMETER 16
`PARAMETER` 47, 60
PARAMETER attribute **52**
PARAMETER statement **60**
parameter-stmt **60**
parentheses 98
partially [storage] associated **286**
part-ref **75**
`POINTER` **38**, 47, 60
pointer assignment **110**
pointer association **284**
pointer association status **284**
POINTER attribute **57**
POINTER statement **60**
pointer-assignment-stmt **110**
pointer-stmt **60**
`POSITION=` specifier **142**, **159**
positional arguments 217
position-edit-desc **163**
precedence of operators **105**
PRECISION intrinsic 32
preconnected files **139**
prefix **206**
present (dummy argument) **202**
PRESENT intrinsic 57
primary **86**
primary **86**
`PRINT` 144
PRINT statement **144**
printing **153**
print-stmt **144**
`PRIVATE` **38**, 52
PRIVATE attribute **52**
PRIVATE statement 40, **58**, 187
procedure 12
  characteristics of **192**
  dummy **191**
  elemental **213**
  external **191**
  internal **191**
  intrinsic **217–274**
  non-Fortran **211**
  pure **212**
procedure interface **192**
procedure interface block 12
procedure reference 18, **198**
procedure references
  generic **277**
  resolving **278**
processor **1**
`PROGRAM` 185
program 12
program **9**
program name **185**
program unit 11
program-stmt **185**
program-unit **9**

PUBLIC 52
PUBLIC attribute **52**
PUBLIC statement **58**, 187
PURE 207
pure procedure **212**

Q

QUOTE (DELIM value) 183

R

RANGE intrinsic 31, 32
rank 17, **17**
READ 144
READ statement **144**
READ= specifier **159**
read-stmt **144**
READWRITE= specifier **159**
REAL 33, 47
real and complex editing **166**
real model **219**
real part **34**
real type **32–34**
REAL type specifier **50**
real-literal-constant **33**
REC= specifier **146**
RECL= specifier **141, 158**
record **133**
RECURSIVE 207
relational intrinsic operations **102**
rename **187**
repeat specification **162**
resolving procedure references **278**
restricted expression **95**
RESULT 206, 209
result variable 12
RETURN 210
RETURN statement **210**
return-stmt **210**
REWIND 154
REWIND statement **155**
rewind-stmt **154**

S

SAVE 47, 59
SAVE attribute **56**, 61
SAVE statement **59**
saved object **56**
save-stmt **59**
scalar **17, 74**
scale factor **163**
scope of names **275**
scoping unit 11
section-subscript **76**
SELECT CASE 123
SELECT CASE statement **123**
select-case-stmt **123**
SELECTED_INT_KIND intrinsic 31, 218
SELECTED_REAL_KIND intrinsic 32, 218
SEQUENCE **38**
sequence association **202**
SEQUENCE property 43
SEQUENCE statement 38, 39
sequence structure **52**
sequence type **38, 39**
sequential access **135**
sequential access input/output statement **146**
SEQUENTIAL= specifier **157**
shape **17**
signed-int-literal-constant **31**
sign-edit-desc **163**
size **17**
SIZE= specifier **147**
specific interface **194**
specification expression **95**
specification function 49, **96**
specification-expr **95**
specifications **47–71**
specification-stmt **10**
standard-conforming program **2**
STAT= 79
statement 24
statement function **211**
statement keyword 18
statement label **24**, 130
statement order **13**
statements
    accessibility **58**
    ALLOCATABLE **60**
    ALLOCATE **79**
    arithemetic IF **131**
    assignment **107**
    attribute specification **57–71**
    BACKSPACE **154**
    CALL **198**
    CASE **123**
    CLOSE **143**
    COMMON **68–71**
    computed GO TO **131**
    CONTAINS **211**
    CONTINUE **131**
    CYCLE **129**

  DATA **61**
  data transfer **144**
  DEALLOCATE **82**
  DIMENSION **59**
  direct access input/output **146**
  DO **126**
  DO WHILE **126**
  END **14**
  ENDFILE **155**
  EQUIVALENCE **66–68**
  EXIT **129**
  EXTERNAL **197**
  file positioning **154**
  FORALL **114**, **118**
  FORMAT **161**
  formatted input/output **145**
  FUNCTION **206**
  GO TO **131**
  IF **123**
  IMPLICIT **63**
  input/output **133–160**
  INQUIRE **155**
  INTENT **58**
  INTRINSIC **197**
  list-directed input/output **146**
  MODULE **186**
  MODULE PROCEDURE **194**
  NAMELIST **65**
  namelist input/output **146**
  NULLIFY **82**
  OPEN 139
  OPTIONAL **58**
  PARAMETER **60**
  POINTER **60**
  PRINT **144**
  PRIVATE **58**
  PROGRAM **185**
  PUBLIC **58**
  READ **144**
  RETURN **210**
  REWIND **155**
  SAVE **59**
  SELECT CASE **123**
  sequential access input/output **146**
  STOP **131**
  SUBROUTINE **208**
  TARGET **60**
  type declaration **47–57**
  unformatted input/output **145**
  WHERE **111**
  WRITE **144**
STATUS= specifier **141**, **143**
stmt-function-stmt **211**
STOP 131
STOP statement **131**
stop-stmt **131**
storage associated **286**
storage association **66–71**, **285**
storage sequence 69, **285**
storage unit **285**
stride **78**
string
  - *see* character string
structure 15, **52**
structure component **75**
structure-constructor **44**
subobject **73**
subobject designator 18
subobjects **16**
SUBROUTINE 208
subroutine 12
subroutine reference **206**
subroutine subprogram **208**
subroutine-stmt **208**
subroutine-subprogram **9**
subscript **76**, **114**
subscript triplet **78**
subscript-triplet **76**
substring **74**

T

TARGET 47, 60
target **110**
TARGET attribute **57**
TARGET statement **60**
target-stmt **60**
terminal point **136**
THEN 122
totally [storage] associated **286**
transfer of control 121, **130**, 147
transformational functions (intrinsic) **217**
TYPE **38**, 47
type
  character **35–37**
  complex **34**
  derived types **37–45**
  integer **31–32**
  intrinsic types **31–37**
  logical **37**
  nonnumeric types **35–**46
  numeric types **31–35**
  real **32–34**
type conformance **108**
type declaration statements **47–57**
type equality **43**

type parameter 31, 32
type specifier **50–52**
  CHARACTER **50**
  COMPLEX **50**
  derived type **52**
  DOUBLE PRECISION **50**
  INTEGER **50**
  LOGICAL **51**
  REAL **50**
  TYPE **52**
TYPE type specifier **52**
type-declaration-stmt **47**
type-spec **47**

U

undefined 18
undefinition of variables **288**
unformatted data transfer **152**
unformatted input/output statement **145**
unformatted record **133**
`UNFORMATTED=` specifier **158**
unit **138**
`USE` 187
use association **282**
USE statement **187**
use-stmt **187**

V

variable **73**
variables **16**
  definition & undefinition **288**
vector subscript **78**

W

`WHERE` 111
WHERE construct **111**
WHERE statement **111**
where-construct **111**
where-stmt **111**
`WHILE` 126
`WRITE` 144
WRITE statement **144**
`WRITE=` specifier **159**
write-stmt **144**

Z

zero-size array 17, **54**, 62
zero-sized arrays 151