Preface

There has been an explosion of interest in, and books on object-oriented programming (OOP). Why have yet another book on the subject? In the past a basic education was said to master the three r's: reading, 'riting, and 'rithmetic. Today a sound education in engineering programming leads to producing code that satisfy the four r's: readability, reusability, reliability, and really-efficient. While some object-oriented programming languages have some of these abilities Fortran 90/95 offers all of them for engineering applications. Thus this book is intended to take a different tack by using the Fortran 90/95 language as its main OOP tool. With more than one hundred pure and hybrid object-oriented languages available, one must be selective in deciding which ones merit the effort of learning to utilize them. There are millions of Fortran programmers, so it is logical to present the hybrid object-oriented features of Fortran 90/95 to them to update and expand their programming skills. This work provides an introduction to Fortran 90 as well as to object-oriented programming concepts. Even with the current release (Fortran 95) we will demonstrate that Fortran offers essentially all of the tools recommended for object-oriented programming techniques. It is expected that Fortran 200X will offer additional object-oriented capabilities, such as declaring "extensible" (or virtual) functions. Thus, it is expected that the tools learned here will be of value far into the future.

It is commonly agreed that the two decades old F77 standard for the language was missing several useful and important concepts of computer science that evolved and were made popular after its release, but it also had a large number of powerful and useful features. The following F90 standard included a large number of improvements that have often been overlooked by many programmers. It is fully compatible with all old F77 standard code, but it declared several features of that standard as obsolete. That was done to encourage programmers to learn better methods, even though the standard still supports those now obsolete language constructs. The F90 standards committee brought into the language most of the best features of other more recent languages like Ada, C, C++, Eiffel, etc. Those additions included in part: structures, dynamic memory management, recursion, pointers (references), and abstract data types along with their supporting tools of encapsulation, inheritance, and the overloading of operators and routines. Equally important for those involved in numerical analysis the F90 standard added several new features for efficient array operations that are very similar to those of the popular MATLAB environment. Most of those features include additional options to employ logical filters on arrays. All of the new array features were intended for use on vector or parallel computers and allow programmers to avoid the bad habit of writing numerous serial loops. The current standard, F95, went on to add more specific parallel array tools, provided "pure" routines for general parallel operations, simplified the use of pointers, and made a few user friendly refinements of some F90 features. Indeed, at this time one can view F90/95 as the only cross-platform international standard language for parallel computing. Thus Fortran continues to be an important programming language that richly rewards the effort of learning to take advantage of its power, clarity, and user friendlyness.

We begin that learning process in Chapter 1 with an overview of general programming techniques. Primarily the older "procedural" approach is discussed there, but the chapter is closed with an outline of the newer "object" approach to programming. An experienced programmer may want to skip directly to the last section of Chapter 1 where we outline some object-oriented methods. In Chapter 2, we introduce the concept of the abstract data types and their extension to classes. Chapter 3 provides a fairly detailed introduction to the concepts and terminology of object-oriented programming. A much larger supporting glossary is provided as an appendix.

For the sake of completeness Chapter 4 introduces language specific details of the topics discussed in

the first chapter. The Fortran 90/95 syntax is used there, but in several cases cross-references are made to similar constructs in the C++ language and the MATLAB environment. While some readers may want to skip Chapter 4, it will help others learn the Fortran 90/95 syntax and/or to read related publications that use C++ or MATLAB. All of the syntax of Fortran 90 is also given in an appendix.

Since many Fortran applications relate to manipulating arrays or doing numerical matrix analysis, Chapter 5 presents a very detailed coverage of the powerful intrinsic features Fortran 90 has added to provide for more efficient operations with arrays. It has been demonstrated in the literature that object-oriented implementations of scientific projects requiring intensive operations with arrays execute much faster in Fortran 90 than in C++. Since Fortran 90 was designed for operations on vector and parallel machines that chapter encourages the programmer to avoid unneeded serial loops and to replace them with more efficient intrinsic array functions. Readers not needing to use numerical matrix analysis may skip Chapter 5.

Chapter 6 returns to object-oriented methods with a more detailed coverage of using object-oriented analysis and object-oriented design to create classes and demonstrates how to implement them as an OOP in Fortran 90. Additional Fortran 90 examples of inheritance and polymorphism are given in Chapter 7. Object-oriented programs often require the objects to be stored in some type of "container" or data structure such as a stack or linked-list. Fortran 90 object-oriented examples of typical containers are given in Chapter 8. Some specialized topics for more advanced users are given in Chapter 9, so beginning programmers could skip it.

To summarize the two optional uses of this text; it is recommended that experienced Fortran programmers wishing to learn to use OOP cover Chapters 2, 3, 6, 7, 8, and 9, while persons studying Fortran for the first time should cover Chapters 1, 2, 3, and. Anyone needing to use numerical matrix analysis should also include Chapter 5.

A OO glossary is included in an appendix to aid in reading this text and the current literature on OOP. Another appendix on Fortran 90 gives an alphabetical listing on its intrinsic routines, a subject based list of them, a detailed syntax of all the F90 statements, and a set of example uses of every statement. Selected solutions for most of the assignments are included in another appendix along with comments on those solutions. The final appendix gives the C++ versions of several of the F90 examples in the text. They are provided as an aid to understanding other OOP literature. Since F90 and MATLAB are so similar the corresponding MATLAB versions often directly follow the F90 examples in the text.

Ed Akin, Rice University, 2002

Index

abstract data type, 15, 23, 27
abstraction, 19, 27
access, 36
access restriction, 19
accessibility, 19
accessor, 18
actual argument, 56
Ada, 33
addition, 56
ADT, *see* abstract data type
ADVANCE specifier, 42, 103
agent, 18
algorithm, 51
ALLOCATABLE, 15
allocatable array, 160, 161
ALLOCATE, 15
allocate, 42
ALLOCATE statement, 75, 93
ALLOCATED, 15
allocation status, 75
AND operand, 42
area, 34
argument
    inout, 71
    input, 71
    interface, 76
    none, 71
    number of, 76
    optional, 76, 77
    order, 76
    output, 71
    rank, 76
    returned value, 76
    type, 76
array, 26, 60, 67, 83
    allocatable, 160
    assumed shape, 77
    automatic, 90, 160
    Boolean, 168
    constant, 160
    dummy dimension, 160
    flip, 170
    mask, 168, 183
    rank, 77, 159, 161, 170
    rectangular, 170
    reshape, 159
    shape, 159
    shift, 172
    size, 159
    unknown size, 77
    variable rank, 160
array operations, 163
ASCII, 23
ASCII character set, 77, 78, 99, 163
assembly language, 15
assignment operator, 10, 39
ASSOCIATED, 15
ASSOCIATED function, 76, 89
ASSOCIATED intrinsic, 132, 134
associative, 176, 177
asterisk (*), 58
ATAN2, 13
attribute, 105, 106, 109, 121, 125
    private, 27, 125
    public, 27
    terminator, 25
attribute terminator, 25
attributes, 19, 27
automatic array, 90, 160, 161
automatic deallocation, 29

BACKSPACE statement, 76
bad style, 162
base class, 121
behavior, 106, 109
binary file, 163
bit
    clear, 75
    extract, 75
    set, 75
    shift, 75
    test, 75
bit manipulation, 75
blanks
    all, 78
    leading, 78
    trailing, 78
Boolean, 53
Boolean value, 23

bottom-up, 4
bounds, 159
bubble sort, 93, 95
    ordered, 96
bug, 9

C, 1, 33, 52
C++, 1, 10, 14, 24, 33, 52, 58, 60, 77, 82, 103, 123
CALL statement, 42
CASE DEFAULT statement, 64
CASE statement, 64
cases, 62
central processor unit, 73
character, 82
    case change, 81
    control, 77
    from number, 81
    functions, 78
    non-print, 103
    non-printable, 77
    strings, 77
    to number, 81
character set, 23
CHARACTER type, 23, 26, 53
chemical element, 25
circuits, 170
circular shift, 172
class, 15, 19, 33
    base, 18
    Date, 120, 123
    derived, 18
    Drill, 105
    Employee, 125
    Geometric, 120
    Global_Position, 114
    Great_Arc, 114
    hierarchy, 33
    instance, 33
    Manager, 125, 135
    Person, 120, 123
    polymorphic, 133
    Position_Angle, 109, 114
    Professor, 123
    Student, 120, 123
class code
    class_Angle, 114
    class_Circle, 34
    class_Date, 37
    class_Fibonacci_Number, 29
    class_Person, 37
    class_Rational, 42
    class_Rectangle, 34
    class_Student, 37
    Drill, 106
    Global_Position, 114
    Great_Arc, 114
    Position_Angle, 114
clipping function, 14, 71
CLOSE statement, 75
Coad/Yourdon method, 18
colon operator, 56, 61, 62, 78, 160, 163, 167, 170
column major order, 181
column matrix, 174
column order, 162
comma, 99
comment, 1, 2, 7, 9, 12, 52
commutative, 101, 176, 177
compiler, 10, 15, 91
complex, 10, 82, 165
COMPLEX type, 23, 53
COMPLEX type , 24
composition, 34, 36
conditional, 7–9, 11, 51, 58
conformable, 176
connectivity, 170
constant array, 160
constructor, 18, 29, 34, 125, 134, 135
    default, 18
    intrinsic, 18, 26, 34, 39
    manual, 36
    public, 37
    structure, 26
CONTAINS statement, 29, 33, 34, 73, 76, 86
continuation marker, 10
control key, 79
conversion factors, 29
count-controlled DO, 12, 13
CPU, *see* central processor unit
curve fit, 91
CYCLE statement, 66

data abstraction, 19
data hiding, 36
data types, 10
    intrinsic, 23
    user defined, 23
date, 101
DEALLOCATE, 15
deallocate, 18, 42
DEALLOCATE statement, 75
debugger, 17
debugging, 16
default case, 64
default value, 29
dereference, 58
derived class, 121

derived type, 15, 23
component, 83
nested, 83
print, 85
read, 85
destructor, 29, 34, 41, 48
determinant, 179
diagonal matrix, 174
dimension
constant, 161
extent, 159
lower bound, 159
upper bound, 159
distributive, 177
division, 56
DO statement, 29, 60, 62
DO WHILE statement, 67
DO-EXIT pair, 68, 69
documentation, 17
domain, 19
dot_product, 12
double, 24
DOUBLE PRECISION type, 23, 24, 53
dummy argument, 56, 73
dummy dimension, 161
dummy dimension array, 160
dummy variable, 73
dynamic binding, 18
dynamic data structures, 38
dynamic dispatching, 132
dynamic memory, 75
allocation, 15
de-allocation, 15
management, 15
dynamic memory management, 89

e, 25
EBCDIC, 23
EBCDIC character set, 77
Eiffel, 18
electric drill, 105
ELSE statement, 42, 63, 67
encapsulate, 15
encapsulation, 27, 33
end off shift, 172
end-of-file, 76
end-of-record, 76
end-of-transmission, 78
EOF, *see* end-of-file
EOR, *see* end-of-record
EOT, *see* end of transmission
equation
number, 173
error checking, 18
exception, 75
exception handler, 75
exception handling, 18
exercises, 48
EXIT statement, 66, 67
explicit loop, 11
exponent range, 24
exponentiation, 56
expression, 10, 51, 52, 89
external subprogram, 77

factorization, 178, 179
FALSE result, 63
Fibonacci number, 29
file, 75
column count, 100
internal, 81
line count, 100
read status, 100
unit number, 101
finite element, 43
flip, 167, 170
float, 53
floating point, *see* real, 23, 24, 183
flow control, 11, 51, 58
FORMAT statement, 34
function, 7, 9, 51, 69, 70
argument, 13, 15
extensible, 132
recursive, 42, 102
result, 70
return, 13
variable, 15
function code
Add, 29
add_Rational, 42
Angle_, 114
circle_area, 34
clip, 71
convert, 42
copy_Rational, 42
Date_, 37
Decimal_min, 114
Decimal_sec, 114
Default_Angle, 114
Drill_, 106
gcd, 42, 102
get_Arc, 114
get_Denominator, 42
get_Latitude, 114
get_Longitude, 114
get_mr_rate, 106
get_next_io_unit, 103
Get_Next_Unit, 99

get_Numerator, 42
get_person, 37
get_torque, 106
Great_Arc_, 114
inputCount, 93
Int_deg, 114
Int_deg_min, 114
Int_deg_min_sec, 114
is_equal_to, 42
make_Person, 37
make_Rational, 42
make_Rectangle, 36
make_Student, 37
mid_value, 70
mult_Fraction, 87
mult_Rational, 42
new_Fibonacci_Number, 29
Person_, 37
Rational_, 42
rectangle_area, 34
set_Date, 37
set_Lat_and_Long_at, 114
Student_, 37
toc, 73
to_Decimal_Degrees, 114
to_lower, 81
to_Radians, 114
to_upper, 81, 101
FUNCTION statement, 29

Game of Life, 4
Gamma, 25
gather-scatte, 172
gcd, *see* greatest common divisor
generic function, 33, 34
generic interface, 134
generic name, 34
generic object, 42
generic routine, 123
generic subprogram, 77
geometric shape, 34
global positioning satellite, 108
global variable, 14, 73
GO TO statement, 65, 66
GPS, *see* global positioning satellite
Graham method, 18
graphical representation, 27, 120
greatest common divisor, 42, 102

Has-A, 109
header file, 131
hello world, 52, 101
hierarchie
kind of, 18
part of, 18
horizontal tab, 78
Hubbard, J.R., 36
hyperbolic tangent, 103

identity matrix, 182
if, 12
IF ELSE statement, 62
IF statement, 29, 37, 42, 62
if-else, 12
IF-ELSE pair, 63
IF-ELSEIF, 132
IMPLICIT COMPLEX, 53
IMPLICIT DOUBLE PRECISION, 53
IMPLICIT INTEGER, 52
implicit loop, 12
IMPLICIT NONE, 26, 29
IMPLICIT REAL, 52
implied loop, 61, 62, 160, 170
INCLUDE line, 37, 42, 90
INDEX intrinsic, 81
indexed loop, 11
infinite loop, 9, 68, 69
inheritance, 18, 33, 34, 73, 121
inherited, 37
inner loop, 62
INQUIRE intrinsic, 93, 98, 103
INQUIRE statement, 76
instance, 33, 124
integer, 10, 82, 165
INTEGER type, 23, 24, 53
intent
in, 29
inout, 29
statement, 29
INTENT statement, 29, 58, 71, 94
interface, 2, 6, 9, 13, 15, 27, 34, 76, 93, 106, 109, 123
general form, 77
human, 18
input/output, 18
prototype, 18
INTERFACE ASSIGNMENT (=) block, 87
interface block, 34, 77
interface body, 77
interface operator (*), 39
INTERFACE OPERATOR block, 86, 87
INTERFACE OPERATOR statement, 170
interface prototype, 105, 106, 125
INTERFACE statement, 34
internal file, 81
internal sub-programs, 73
interpreter, 10, 15
intrinsic, 170

intrinsic constructor, 86, 99, 108
intrinsic function, 12, 70
inverse, 182
IOSTAT = variable, 75, 76
Is-A, 108, 109
ISO_VARIABLE_LENGTH_STRING, 23
Is_A, 126

keyword, 123
KIND intrinsic, 24
Kind-Of, 109, 125

latitude, 108
least squares, 91
LEN intrinsic, 78, 81
length
    line, 52
    name, 52
LEN_TRIM intrinsic, 78
lexical operator, 95
lexically
    greater than, 78
    less than, 78
    less than or equal, 78
library function, 16
line continuation, 101
linear equations, 177, 178
linked list, 38, 88, 89
linker, 16, 90
list
    doubly-linked, 89
    singly-linked, 89
logarithm, 70, 92
logical, 82
    AND, 63
    equal to, 63
    EQV, 63
    greater than, 63
    less than, 63
    NEQV, 63
    NOT, 63
    operator, 63
    OR, 63
logical expression, 11
logical mask, 62
logical operator, 63
LOGICAL type, 23, 42
long, 24
long double, 24
long int, 24
longitude, 108
loop, 5, 7–9, 11, 51, 58, 183
    abort, 68
    breakout, 66
    counter, 60
    cycle, 66
    exit, 60, 66
    explicit, 59
    implied, 61
    index, 101
    infinite, 60, 68
    nested, 62, 66
    pseudocode, 59
    skip, 66
    until, 67, 68
    variable, 60
    while, 67
loop control, 61, 162
loop index, 101
loop variable, 11
lower triangle, 175, 178

manual constructor, 86, 106
manual page, 17
mask, 165, 168, 169, 183
masks, 62
Mathematica, 51
mathematical constants, 25
Matlab, 1, 10, 14, 52, 61, 70, 100, 103
MATMUL intrinsic, 177
matrix, 159, 174
    addition, 176
    algebra, 159
    column, 174
    compatible, 176
    determinant, 179
    diagonal, 174
    factorization, 178
    flip, 167
    identity, 178
    inverse, 90, 178
    multiplication, 163, 176
    non-singular, 178
    null, 174
    skew symmetric, 175
    solve, 90
    square, 174, 175
    symmetric, 175
    Toeplitz, 175
    transpose, 163, 175
    triangular, 175, 178
matrix addition, 181, 182
matrix algebra, 159, 176
matrix multiplication, 169, 177, 182
matrix operator, 38
matrix transpose, 169
maximum values, 71
MAXLOC intrinsic, 71

MAXVAL intrinsic, 71
mean, 70
member, 121
message, 27
methods, 3
    private, 27
    public, 27
military standards, 75
minimum values, 71
MINLOC intrinsic, 71
MINVAL intrinsic, 71
modular design, 6
module, 15, 25, 33, 69
module code
    class_Angle, 114
    class_Circle, 34
    class_Date, 37
    class_Fibonacci_Number, 29
    class_Global_Position, 114
    class_Great_Arc, 114
    class_Person, 37
    class_Position_Angle, 114
    class_Rational, 42
    class_Rectangle, 34
    class_Student, 37
    exceptions, 76
    Fractions, 87
    Math_Constants, 25
    record_Module, 97
    tic_toc, 73, 101
MODULE PROCEDURE statement, 34, 39, 86, 87, 170
MODULE statement, 29
module variable, 29
modulo function, 56
multiple inheritanc, 121
multiplication, 56
Myer, B., 18

NAG, *see* National Algorithms Group
named
    CYCLE, 66, 67
    DO, 67
    DO loop, 60
    EXIT, 66, 67
    IF, 64
    SELECT CASE, 64
National Algorithms Group, 91
nested
    DO, 67
    IF, 62
new line, 79, 103
Newton-Raphson method, 11
non-advancing I/O, 42
NULL function (f95), 89
NULLIFY, 15
nullify, 134
NULLIFY statement, 89
number
    bit width, 24
    common range, 24
    label, 60
    significant digits, 24
    truncating, 166
    type, 24
numeric types, 23
numerical computation, 38

object, 15, 19, 33
object oriented
    analysis, 18, 43, 105, 109, 120
    approach, 18
    design, 18, 43, 105, 109, 120
    language, 18
    programming, 18, 105
    representation, 18
Object Pascal, 18
OOA, *see* object oriented analysis
OOD, *see* object oriented design
OOP, *see* object oriented programming
OPEN statement, 75, 163
operator, 27
    .op., 87, 169
    .solve., 90, 91
    .t., 170
    .x., 170
    assignment, 39
    binary, 87
    defined, 18, 87
    extended, 87
    overloaded, 18
    overloading, 39, 86
    symbol, 87
    unary, 87
    user defined, 77, 169
operator overloading, 10
operator precedence, 52
operator symbol, 169
optional argument, 29, 37, 76
OPTIONAL attribute, 29, 36, 106
OR operand, 37
ordering array, 96
outer loop, 62
overloaded member, 123
overloading, 39, 48, 86
    testing, 87

package, 15

parallel computer, 43
PARAMETER attribute, 25
Part-Of, 109
partial derivative, 180
partitioned matrix, 175
pass by reference, 57, 77, 88
pass by value, 57, 58, 77
path name, 37
pi, 25
pointer, 10, 23, 76, 87
    allocatable, 15
    arithmetic, 88
    assignment, 89
    association, 88
    declaration, 88
    dereference, 58
    detrimental effect, 88
    in expression, 89
    inquiry, 89
    nullify, 89
    status, 15, 88
    target, 88
pointer object, 133
pointer variable, 87
polymorphic class, 133
polymorphic interface, 120
polymorphism, 18, 33, 34, 121, 126
portability, 15
pre-processor, 131
precedence rules, 11
precision, 183
    double, 82
    kind, 24
    portable, 82
    single, 82
    specified, 82
    underscore, 24
    user defined, 24
precision kind, 24
PRESENT function, 76
PRESENT intrinsic, 29, 36
PRINT * statement, 29
private, 33, 106
PRIVATE attribute, 29, 36
private attributes, 37
PRIVATE statement, 27
procedural programming, 18
procedure, 69
program
    documentation, 17
    executable, 17
    scope, 14
program code, 114
    array_indexing, 60
    clip_an_array, 71
    create_a_type, 26
    declare_interface, 77
    Fibonacci, 29
    geometry, 34
    if_else_logic, 63
    linear_fit, 93
    Logical_operators, 63
    main, 37, 42
    operate_on_strings, 79
    relational_operators, 63
    simple_loop, 60
    string_to_numbers, 81
    structure_components, 85
    test_bubble, 98
    test_Drill, 108
    test_Fractions, 87
    test_Great_Arc, 114
program keyword, 56
PROGRAM statement, 26, 29
projectile, 102
prototype, 6, 76
pseudo-pointer, 96
pseudocode, 5, 14, 51, 71, 102
public, 33, 125
PUBLIC attribute, 29
public constructor, 37
public method, 27
PUBLIC statement, 27

quadratic equation, 3
queue, 89

rank, 161
rational number, 38, 39
read error, 103
READ statement, 29, 62, 76
real, 10, 82, 165
REAL type, 23, 24, 53
recursive algorithm, 88
RECURSIVE qualifier, 42, 102
reference, 10
relational operator, 52, 63, 78
remainder, 56
rename modifier, 121
reshape, 162
RESULT option, 29
result value, 70
return, 161
RETURN statement, 66
REWIND statement, 76

sample data, 99

scatter, 173
scope, 14
SELECT CASE statement1, 64
SELECTED_INT_KIND, 23, 24
SELECTED_REAL_KIND, 23, 24
selector symbol, 26, 29, 34
server, 18
short, 24
size, 12
SIZE intrinsic, 70, 90, 93, 159
Smalltalk, 18
sort, 87, 91, 93, 96, 127
    bubble, 93
    characters, 95
    object, 97
    objects, 95
    strings, 95
sorting, 42
sparse vector, 49
sparse vector class, 183
specification, 4
SQRT intrinsic, 27
square root, 27, 56, 70
stack, 89
STAT = variable, 75
statement, 2, 9
statement block, 12, 58
statements, 1
status
    FILE, 76
    IOSTAT =, 76
    MODE, 76
    OPENED =, 76
status checking, 161
STOP statement, 37
storage
    column wise, 159
    row wise, 159
string, 23, 56
    adjust, 78
    case change, 81
    character number, 78
    collating sets, 78
    colon operator, 78
    concatenate, 78
    copy, 78
    dynamic length, 77
    from number, 81
    functions, 78
    length, 78
    logic, 78
    repeat, 78
    scan, 78
    to number, 81
    trim, 78
    verify, 78
strings, 77
strong typing, 53
struct, 53
structure, 23, 25, 33, 85
structure constructor, 26
structured programming, 13
submatrix, 175
subprogram, 69
    recursive, 102
subroutine, 69, 70
subroutine code, 114
    assign, 87
    Change, 77
    delete_Rational, 42
    equal_Fraction, 87
    equal_Integer, 42
    exception_status, 76
    in, 106
    Integer_Sort, 96, 99
    invert, 42
    list, 42
    List_Angle, 114
    List_Great_Arc, 114
    List_Position, 114
    List_Position_Angle, 114
    List_Pt_to_Pt, 114
    lsq_fit, 93
    mult_Fraction, 87
    No_Change, 77
    out, 106
    Print, 29
    print_Date, 37
    print_DOB, 37
    print_DOD, 37
    print_DOM, 37
    print_GPA, 37
    print_Name, 37
    print_Sex, 37
    readData, 93, 101
    read_Date, 37
    Read_Position_Angle, 114
    reduce, 42
    set_DOB, 37
    set_DOD, 37
    set_DOM, 37
    set_Latitude, 114
    set_Longitude, 114
    simple_arithmetic, 56
    Sort_Reals, 94
    Sort_String, 95

String_Sort, 99
test_matrix, 90
tic, 73
SUBROUTINE statement, 29
subroutines, 33
subscrip, 159
subscript, 26, 60
bounds, 159
range, 181
vector, 170
subtraction, 56
subtype, 133
subtyping, 126, 132
sum, 12
SUM intrinsic, 70, 93, 169
super class, 121
syntactic error, 17
SYSTEM_CLOCK intrinsic, 73

tab, 79, 99, 103
TARGET, 15
target, 23, 76, 88, 89
template, 43, 126, 128
tensor, 159
testing, 15
time of day, 101
Toeplitz matrix, 175
top-down, 4
transformational functions, 169
transpose, 163, 175, 177
TRANSPOSE intrinsic, 170
tree structure, 38, 88, 89
triplet, *see* colon operator
true, 12
TRUE result, 63
truss, 170
type
conversion, 81
default, 52
implicit, 52
TYPE declaration, 26, 29
TYPE statement, 27, 34

unexpected result, 169
upper triangle, 175, 178
USE association, 121, 125
USE statement, 29, 33, 34, 37, 86, 90
user defined operator, 169
user interface, 2

validation, 29
variable, 8, 10, 23, 51
global, 14
name, 10
type, 10
variable rank array, 160
vector, 159
vector class, 48, 183
vector subscript, 62, 170
volume, 48

WHERE construct, 169
WHERE statement, 62, 67, 169
while-true, 68
wildcard, 128
WRITE statement, 34, 62, 76