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 friendliness.

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

Acknowledgements

We are all indebted to the hundreds of programmers that labor on various standards committees to continually improve all programming languages. Chapter 1 is a modification of introductory programming notes developed jointly with Prof. Don Johnson at Rice University. I would like to thank Prof. Tinsley Oden and the Texas Institute for Computational Mathematics for generously hosting my sabbatical leave where most of this work was developed, and Rice University for financing the sabbatical. Special thanks go to my wife, Kimberly, without whose support and infinite patience this book would not have been completed.

Source Codes

All of the example programs and selected solutions are included on the CD-ROM provide with the book. To be readable on various platforms they have been written with the ISO9660 standard format. Additional files are provided to relate the ISO standard short filenames to the full length program names used in the book. Of course, the source files will have to be processed through a Fortran 90 or 95 or 2000 compiler to form executables. All of the figures are also provided as encapsulated Postscript (tm) files.

Index

−, 53, 56
<=, 53
>=, 53
\, 122
*, 10, 56
**, 56
+, 53, 56
/, 10, 56
::, 25, 53
=, 10
=>, 143
%, 51, 143
&, 10, 34, 37, 42
/=, 53
==, 53
=>, 121

ABS function, 56, 162, 250
absolute value, 56, 162
abstract class, 285
abstract data type, 15, 23, 27, 285
abstraction, 19, 27, 285
access, 36
access operation, 142
access restriction, 19
accessibility, 19
accessor, 18, 285
ACHAR function, 77, 80
ACOS function, 56, 162
actual argument, 56
Ada, 15, 33
addition, 56
ADJUSTL function, 77
ADJUSTR function, 77
ADT, *see* abstract data type
ADVANCE specifier, 42, 102
agent, 18
AIMAG function, 56, 162
AINT function, 56, 162
algorithm, 51
ALL function, 162, 255
all mask elements true, 162
allocatable array, 156, 157, 285
ALLOCATABLE attribute, 183
ALLOCATABLE statement, 15
allocate, 42
ALLOCATE statement, 15, 74, 92, 181, 183
ALLOCATED function, 15, 181, 183
allocation status, 74, 181, 258
AND operand, 42, 63, 104
AND operator, 53
ANINT function, 162
ANY function, 162, 181
any mask element true, 162
arc cosine, 56
arc sine, 56
arc tangent, 56
arccosine, 162
arcsine, 162
arctangent, 162
arctangent for complex number, 162
area, 34
argument, 285
  inout, 69
  input, 69
  interface, 75
  none, 69
  number of, 75
  optional, 75, 76
  order, 75
  output, 69
  rank, 75
  returned value, 75
  type, 75
array, 26, 60, 66, 82, 135, 149, 285
  allocatable, 156
  assumed shape, 76
  automatic, 89, 156
  Boolean, 164
  constant, 156
  dummy dimension, 156
  flip, 166
  mask, 164, 179
  of pointers, 135
  rank, 76, 155, 157, 166
  rectangular, 166
  reshape, 155
  shape, 155
  shift, 168

size, 155
total, 162
unknown size, 76
variable rank, 156
array operations, 159
array pointer, 285
array shape vector, 162
ASCII character set, 23, 76, 77, 98, 159
ASIN function, 56, 162
assembly language, 15
assignment operator, 10, 39, 189, 285
assignment statement, 285
ASSOCIATED function, 15, 75, 88, 130, 132, 181
association, 285
associative, 172, 173
asterisk (*), 58
ATAN function, 56, 162
ATAN2 function, 13, 56, 162
attribute, 103, 104, 107, 119, 123, 192, 285
name, 19
private, 27, 123
public, 27
terminator, 25
attribute terminator, 25
attributes, 19, 27
automatic array, 89, 156, 157, 285
automatic deallocation, 29

BACKSPACE statement, 75
bad style, 158
base 10 logarithm, 56, 162
base class, 119, 286
behavior, 104, 107
binary file, 159
binary operator, 286
binary read, 268
binary write, 183
bit
clear, 74
extract, 74
set, 74
shift, 74
test, 74
bit function
BIT_SIZE, 74
BTEST, 74
IAND, 74
IBCLR, 74
IBITS, 74
IBSET, 74
IEOR, 74
IOR, 74
ISHFT, 74
ISHFTC, 74
MVBITS, 74
NOT, 74
TRANSFER, 74
bit manipulation, 74
blanks
all, 77
leading, 77
trailing, 77
Boolean type, 53
Boolean value, 23
bottom-up, 4
boundary condition, 192
bounds, 155
bubble sort, 92, 94
ordered, 95
bug, 9

C, 1, 33, 52
C++, 1, 10, 14, 24, 33, 52, 58, 59, 76, 81, 102, 121
call by reference, 286
call by value, 286
CALL statement, 42, 76, 86, 89, 92, 97, 121, 123, 124, 131, 137, 140, 142, 143, 149
CASE DEFAULT statement, 63, 188
CASE statement, 63, 188, 272
cases, 62
CEILING function, 56, 162
central processor unit, 72
CHAR function, 77
character, 81
case change, 80
control, 76
from number, 80
functions, 77
non-print, 76, 102
strings, 76
to number, 80
character set, 23
CHARACTER type, 23, 26, 53
chemical element, 25
chemical_element, 128
circuits, 166
circular shift, 168
circular-linked list, 185
class, 15, 19, 33, 286
base, 18
Date, 118, 121
derived, 18
Drill, 103
Employee, 123
Geometric, 118

  - Global_Position, 112
  - Great_Arc, 112
  - hierarchy, 33
  - instance, 33
  - iterator, 192
  - Manager, 123, 133
  - Person, 118, 121
  - polymorphic, 131
  - Position_Angle, 107, 112
  - Professor, 121
  - sparse vector, 258
  - Student, 118, 121
- class attribute, 286
- class code
  - class_Angle, 112
  - class_Circle, 34
  - class_Date, 37
  - class_Employee_1, 122
  - class_Employee_2, 123
  - class_Employee_3, 124
  - class_Fibonacci_Number, 29
  - class_Manager_1, 123
  - class_Manager_2, 123
  - class_Manager_3, 124
  - class_Object, 143
  - class_Person, 37
  - class_Position_Angle, 270
  - class_Professor, 121
  - class_Queue, 140
  - class_Rational, 42
  - class_Rectangle, 34
  - class_sparse_Vector, 258
  - class_Stack, 137
  - class_Student, 37
  - class_Vector, 257
  - Drill, 104
  - elem_type_data_class, 181
  - Global_Position, 112
  - Great_Arc, 112
  - Is_A_Member_Class, 131
  - Member_1_Class, 131
  - Member_2_Class, 131
  - Position_Angle, 112
- class descriptor, 286
- class inheritance, 286
- clipping function, 14, 69
- CLOSE statement, 74, 92, 97, 271
- CMPLX function, 162
- Coad/Yourdon method, 18
- code reuse, 194
- colon operator, 56, 60, 61, 77, 156, 159, 163, 166, 267
  - syntax, 56
- column major order, 177
- column matrix, 170
- column order, 158
- comma, 98
- comment, 1, 2, 7, 9, 12, 51, 52
- commutative, 100, 172, 173
- compiler, 10, 15, 90
- complex, 10, 81, 161
- complex conjugate, 56
- COMPLEX type, 23, 24, 53
- component
  - assignment, 82
  - declaring, 82
  - initializing, 82
  - interpretation, 82
  - referencing, 82
  - syntax, 82
- component selector, 34, 37, 42
- composition, 34, 36, 190, 194
- concatanate, 122
- conditional, 7–9, 11, 51, 58
- conformable, 172
- CONJG function, 56, 162
- conjugate of complex number, 162
- connectivity, 166
- constant array, 156
- constructor, 18, 29, 34, 123, 132, 133, 136, 149, 255, 286
  - default, 18
  - intrinsic, 18, 26, 34, 39
  - manual, 36
  - public, 37
  - structure, 26
- container, 135
- container class, 286
- CONTAINS statement, 29, 33, 34, 72, 75, 85
- continuation marker, 10
- control key, 78
- conversion factors, 29
- convert real to complex, 162
- convert to integer, 162
- convert to real, 162
- COS function, 56, 162, 249
- COSH function, 56, 162
- cosine, 56, 162
- COUNT function, 162, 259, 263
- count-controlled DO, 12, 13
- CPU, *see* central processor unit
- curve fit, 90
- CYCLE statement, 65, 66, 260, 263

- data abstraction, 19
- data hiding, 36, 286
- data structure, 135

data types, 10
    intrinsic, 23
    user defined, 23
date, 99, 265
DATE_AND_TIME intrinsic, 265
deallocate, 18, 42, 181
DEALLOCATE statement, 15, 74, 183
deallocation, 287
debugger, 17, 287
debugging, 16
declaration statement, 287
default case, 63
default constructor, 287
default value, 29
defined operator, 287
dereference, 58
dereferencing, 287
derived class, 119
derived type, 15, 23, 287
    component, 82
    nested, 82
    print, 84
    read, 84
destructor, 29, 34, 41, 48, 254, 287
determinant, 175
diagonal matrix, 170
dimension
    constant, 157
    extent, 155
    lower bound, 155
    upper bound, 155
distributive, 173
division, 56
division remainder, 56
DO statement, 29, 58, 61
DO WHILE statement, 66
DO-EXIT pair, 67, 68
documentation, 17
domain, 19
dot product, 162
dot_product, 12
DOT_PRODUCT intrinsic, 12, 162
double, 24
DOUBLE PRECISION type, 23, 24, 53
doubly linked list, 149
drop fraction, 56
dummy argument, 57, 72, 287
dummy array, 287
dummy dimension, 157
dummy dimension array, 156
dummy pointer, 287
dummy variable, 72
dynamic binding, 18, 287
dynamic data structures, 38
dynamic dispatching, 130
dynamic memory, 74, 181
    allocation, 15
    de-allocation, 15
    management, 15
dynamic memory management, 88

e, 25
EBCDIC character set, 23, 76
efficiency, 194
Eiffel, 18
electric drill, 103
ELSE statement, 42, 63, 66
encapsulate, 15
encapsulation, 27, 33, 192, 194, 287
end off shift, 168
end-of-file, 75
end-of-record, 75
end-of-transmission, 77
EOF, *see* end-of-file
EOR, *see* end-of-record
EOT, *see* end of transmission
EPSILON function, 162
equation
    number, 169
EQV operator, 53
error checking, 18
exception, 74, 287
exception handler, 74
exception handling, 18
exercises, 21, 31, 48, 99, 118, 132, 154, 178, 195
EXIT statement, 65, 66, 251, 260, 262, 263, 265, 269, 272, 273
EXP function, 56, 162, 250
explicit interface, 288
explicit loop, 11
exponent range, 24
exponential, 56, 162
exponentiation, 56
expression, 10, 51, 52, 88
external
    file, 89
    subprogram, 89
external file, 288
external procedure, 288
external subprogram, 76

factorization, 174, 175, 179
FALSE result, 62
Fibonacci number, 29
file, 74
    access, 151

  binary, 183
  column count, 99
  direct access, 150
  I/O, 151
  internal, 80
  line count, 99
  modify, 151
  random, 151
  random access, 150
  read status, 99
  record number, 150
  scratch, 183
  unit number, 100
FILE= specifier, 271
finite difference method, 179
finite element, 43
finite element analysis, 181
flip, 163, 166
float, 53
floating point, *see* real, 23, 24, 179
FLOOR function, 56, 162
flow control, 11, 51, 58
forever loop, *see* infinite loop
FORM= specifier, 271
FORMAT statement, 34, 112
function, 7, 9, 51, 68
  argument, 13, 15
  extensible, 130
  generic, 183
  INTEGER, 140
  LOGICAL, 137, 140
  recursive, 42, 101
  result, 69
  return, 13
  TYPE, 137, 140
  variable, 15
function code
  Add, 29
  add_Rational, 42
  add_Real_to_Vector, 253
  add_Vector, 253
  Angle_, 112
  assign, 253
  circle_area, 34
  clip, 69
  convert, 42
  copy_Rational, 42
  copy_Vector, 254
  Create_Q, 140
  Date_, 37
  Decimal_min, 112
  Decimal_sec, 112
  Default_Angle, 112
dot_Vector, 255, 259
Drill_, 104, 106
D_L_new, 149
el_by_el_Mult, 259
equality_operator_point, 188
equal_to_Object, 143
gcd, 42, 101
getEmployee, 123, 124
getName, 123
getNameE, 122, 124
getNameM, 123, 124
getRate, 122, 124
GetX, 188
GetY, 188
get_Arc, 112
Get_Capacity_of_Q, 140
get_Denominator, 42
get_element, 260
Get_Front_of_Q, 140
get_item_cost, 264
get_item_count, 264
get_item_delay, 264
get_item_name, 264
get_Latitude, 112
Get_Length_of_Q, 140, 142
get_Longitude, 112
get_menu, 273
get_mr_rate, 104
get_next_io_unit, 102, 269
Get_Next_Unit, 98
get_Numerator, 42
Get_Obj_at_Ptr, 149
get_Person, 37
get_person, 37
Get_Ptr_to_Obj, 149
get_torque, 104
Global_Position_, 112
Great_Arc_, 112
initialize_item, 264
inputCount, 92, 265
Int_deg, 112
Int_deg_min, 112
Int_deg_min_sec, 112
is_equal_to, 42, 255, 260
is_item_empty, 264
Is_Q_Empty, 140
is_Q_Empty, 142
Is_Q_Full, 140
is_Q_Full, 142
is_Stack_Empty, 137
is_Stack_Full, 137
is_S_L_empty, 143
largest_index, 260

length, 260
lengthnormalize_Vector, 255
less_than_Object, 143
make_Person, 37
make_Professor, 121
make_Rational, 42
make_Rectangle, 36
make_Stack, 137
make_Student, 37
make_Vector, 253
Manager_, 123, 124
maximum, 70
mid_value, 69
mult_Fraction, 86
mult_Rational, 42
new_Fibonacci_Number, 29
next_generation, 251
norm, 262
normalize_Vecto, 262
pay, 123
payE, 122, 124
payM, 123, 124
Person, 121
Person_, 37
pop_from_Stack, 137
print, 121
Professor, 121
Rational, 42
Rational_, 42
real_mult_Sparse, 262
real_mult_Vector, 255
rectangle_area, 34
rows_of, 262
setDataE, 122, 124
setDataM, 123, 124
set_Date, 37
set_Lat_and_Long_at, 112
size_of, 262
size_Vector, 255
Sparse_mult_real, 262
Student, 37, 121
Student_, 37
subtract_Real, 255
subtract_Vector, 255
Sub_Sparse_Vectors, 263
Sum_Sparse_Vectors, 263
S_L_new, 143
toc, 72
to_Decimal_Degrees, 112
to_lower, 80
to_Radians, 112
to_upper, 80, 100, 266
values, 255
values_of, 263
Vector_, 255
Vector_max_value, 255, 263
Vector_min_value, 255, 263
Vector_mult_real, 255
Vector_To_Sparse, 263
zero_sparse, 263
function definition, 288
FUNCTION statement, 29

Game of Life, 4
Gamma, 25
gather-scatter, 168
gcd, *see* greatest common divisor
generic function, 33, 34, 183, 288
generic interface, 132
generic linked list, 149
generic name, 34
generic object, 42
generic operator, 288
generic routine, 121
generic subprogram, 76
geometric shape, 34
global positioning satellite, 106
global variable, 14, 72
GO TO statement, 64, 65
GPS, *see* global positioning satellite
Graham method, 18
graphical representation, 27, 118
greatest common divisor, 42, 101
greatest integer, 162
grid, 190

Has-A, 107, 194
header file, 129
heat transfer, 185
Hello world, 7
hello world, 52, 100
hierarchie
kind of, 18
part of, 18
High Performance Fortran, 195
horizontal tab, 77
host association, 288
Hubbard, J.R., 36
HUGE function, 162
hyperbolic cosine, 56, 162
hyperbolic sine, 56, 162
hyperbolic tangent, 56, 102, 162

I/O, *see* Input-Output
IACHAR function, 77, 80
ICHAR function, 77
identity matrix, 178

IF, 62
  nested, 62
if, 12
IF ELSE statement, 62
IF statement, 29, 37, 42, 62
if-else, 12
IF-ELSE pair, 63
IF-ELSEIF, 130
imaginary part, 56, 162
IMPLICIT COMPLEX, 53
IMPLICIT DOUBLE PRECISION, 53
IMPLICIT INTEGER, 52
implicit loop, 12
IMPLICIT NONE, 26, 29
IMPLICIT REAL, 52
implied loop, 60, 61, 156, 166
INCLUDE line, 37, 42, 89
INDEX function, 77, 80, 266, 273
indexed loop, 11
infinite loop, 9, 68, 269
information hiding, 288
inheritance, 18, 33, 34, 72, 119, 190, 193, 194, 288
  rename, 119
  selective, 119
inherited, 37
initialize random number, 162
inner loop, 61
INQUIRE intrinsic, 92, 97, 102, 268, 269
INQUIRE statement, 75
instance, 33, 122, 288
INT function, 162
integer, 10, 81, 161
integer nearest to real, 162
INTEGER type, 23, 24, 53
intent, 289
  in, 29, 100
  inout, 29
  out, 100
  statement, 29
INTENT attribute, 142
INTENT statement, 29, 58, 69, 93
interface, 2, 6, 9, 13, 15, 27, 34, 75, 92, 104, 107, 121, 136, 189, 258, 289
  general form, 76
  human, 18
  input/output, 18
  prototype, 18
interface assignment, 258
INTERFACE ASSIGNMENT (=) block, 86
interface block, 34, 76
interface body, 76
interface code
  Add_to_Q, 140
  assign, 131
  Create_Q, 140
  display, 131
  getName, 124
  Get_Capacity_of_Q, 140
  Get_Front_of_Q, 140
  Get_Length_of_Q, 140
  Init, 188, 190
  Is_Q_Empty, 140
  Is_Q_Full, 140
  is_Stack_Empty, 136
  is_Stack_Full, 136
  make_Stack, 136
  MyPrint, 188
  new, 131
  orthonormal_basis, 257
  pop_from_Stack, 136
  Position_Angle_, 270
  PrintPay, 123, 124
  push_on_Stack, 136
  Remove_from_Q, 140
  Set, 188
  swap, 127
  testing_basis, 257
interface operator, 188, 258
interface operator (<), 143
interface operator (*), 39
interface operator (==), 143
INTERFACE OPERATOR block, 85, 86
INTERFACE OPERATOR statement, 166
interface prototype, 103, 104, 123
INTERFACE statement, 34
internal file, 80, 289
internal sub-programs, 72
internal subprogram, 251, 289
interpreter, 10, 15
intrinsic, 166
intrinsic constructor, 85, 98, 106, 136, 289
intrinsic function, 12, 68
inverse, 178
IOLENGTH result, 268
IOSTAT= variable, 74, 75, 271
Is-A, 106, 107, 124, 194
ISO_VARIABLE_LENGTH_STRING, 23
iterator, 143, 149, 191, 192, 289

keyword, 121, 289
KIND intrinsic, 24
Kind-Of, 107, 123

largest integer, 56
largest number, 162
latitude, 106

least integer, 162
least squares, 90, 266, 267
LEN function, 77, 80
LEN intrinsic, 77, 80
length
    line, 52
    name, 52
LEN_TRIM function, 77
LEN_TRIM intrinsic, 77
lexical operator, 94
lexically
    greater than, 77
    less than, 77
    less than or equal, 77
LGE function, 77
LGT function, 77
library function, 16
line continuation, 100
linear equations, 173, 174, 179, 184
linked list, 38, 87, 88, 142, 149, 289
    doubly, 149
linked-list, 191
linker, 16, 89, 289
list
    circular, 139, 185, 190
    doubly-linked, 88
    empty, 149
    length, 139
    singly-linked, 88
LLE function, 77
LLT function, 77
local_name, 119
LOG function, 56, 162
LOG10 function, 56, 162
logarithm, 68, 91, 162
logical, 81
    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, 61
LOGICAL type, 23, 42, 137
long, 24
long double, 24
long int, 24
longitude, 106
loop, 5, 7–9, 11, 51, 58, 179
    abort, 66, 67
    breakout, 65
    counter, 59
    cycle, 65, 66
    exit, 59, 65, 66
    explicit, 58
    implied, 60
    index, 100
    infinite, 60, 67, 68
    nested, 61, 65
    pseudocode, 58
    skip, 65
    until, 66, 67
    variable, 60
    while, 66
loop construct, 59
loop control, 60, 158
loop index, 100
loop variable, 11
lower triangle, 171, 174

manual constructor, 85, 104
manual page, 17
mask, 161, 164, 165, 179, 259
masks, 61
Mathematica, 51
mathematical constants, 25
mathematical functions, 56
Matlab, 1, 10, 14, 52, 60, 68, 99, 102
MATMUL intrinsic, 162, 173
matrix, 155, 170, 289
    addition, 172
    algebra, 155
    column, 170
    compatible, 172
    determinant, 175
    diagonal, 170
    factorization, 174
    flip, 163
    identity, 174
    inverse, 89, 174
    multiplication, 159, 172
    non-singular, 174
    null, 170
    skew symmetric, 171
    solve, 89
    sparse, 192
    square, 170, 171
    symmetric, 171
    Toeplitz, 171
    transpose, 159, 171
    triangular, 171, 174
    tridiagonal, 179
matrix addition, 177, 178

- matrix algebra, 155, 172
- matrix multiplication, 162, 165, 173, 178
- matrix operator, 38
- matrix transpose, 162, 165
- maximum array element location, 162
- maximum array element value, 162
- maximum values, 70
- MAXLOC function, 70, 162
- MAXVAL function, 70, 162, 263
- mean, 69
- member, 119
- memory count, 183, 274
- memory leak, 183
- memory management, 181
- message, 27
- message passing, 289
- method, 192, 289
- methods, 3
  - private, 27
  - public, 27
- military standards, 74
- minimum array element location, 162
- minimum array element value, 162
- minimum values, 70
- MINLOC function, 70, 162
- MINVAL function, 70, 162
- MOD function, 56
- modular design, 6
- module, 15, 25, 33, 68, 289
- module code
  - class_Angle, 112
  - class_Circle, 34
  - class_Date, 37
  - class_Employee_1, 122
  - class_Employee_2, 123
  - class_Employee_3, 124
  - class_Fibonacci_Number, 29
  - class_Global_Position, 112
  - class_Great_Arc, 112
  - class_Manager_1, 123
  - class_Manager_2, 123
  - class_Manager_3, 124
  - class_Object, 143
  - class_Person, 37
  - class_Position_Angle, 112, 270
  - class_Professor, 121
  - class_Queue, 140
  - class_Rational, 42
  - class_Rectangle, 34
  - class_sparse_Vector, 258
  - class_Stack, 137
  - class_Student, 37
  - class_Vector, 253, 256, 257
  - Conversion_Constants, 252
  - doubly_linked_list, 149
  - elem_type_data_class, 181
  - exceptions, 75, 137
  - Fractions, 86
  - Gauss_Module, 190
  - inventory_object, 49, 264
  - inventory_system, 270
  - Is_A_Member_Class, 131
  - Math_Constants, 25
  - Member_1_Class, 131
  - Member_2_Class, 131
  - Memory_Status_Count, 183, 274
  - object_type, 136
  - Physical_Constants, 252
  - Point_Module, 188
  - Queue_of_Objects, 140
  - Queue_type, 139
  - record_Module, 97
  - singly_linked_lis, 143
  - singly_linked_list, 143
  - stack_type, 136
  - swap_library, 127
  - tic_toc, 72, 99
- module procedure, 289
- MODULE PROCEDURE statement, 34, 39, 85, 86, 166
- MODULE statement, 29
- module variable, 29
- modulo, 56
- MODULO function, 56
- modulo function, 56
- multiple inheritanc, 119
- multiplication, 56
- Myer, B., 18

- NAG, *see* National Algorithms Group
- named
  - CYCLE, 65, 66
  - DO, 59, 66
  - EXIT, 65, 66
  - IF, 63
  - SELECT CASE, 63
- National Algorithms Group, 90
- natural logarithm, 56
- NEQV operator, 53
- nested, 289
  - DO, 66
  - IF, 62
- new line, 78, 102
- Newton-Raphson method, 11
- NINT function, 56, 162
- node
  - current, 142, 149

dummy, 149
header, 139, 142, 149
linked list, 142
next, 149
null, 142
previous, 142, 149
root, 142
tail, 139
non-advancing I/O, 42
normalized sign, 162
NOT operator, 53
NULL function (f95), 88
nullify, 132
NULLIFY statement, 15, 88, 132
number
bit width, 24
common range, 24
label, 58
significant digits, 24
truncating, 162
type, 24
number of true masks, 162
numberic type, 24
numeric types, 23
numerical computation, 38

object, 15, 19, 33
object oriented
analysis, 18, 43, 103, 107, 118
approach, 18
design, 18, 43, 103, 107, 118, 190
language, 18
programming, 18, 103
representation, 18
Object Pascal, 18
ONLY keyword, 119
OOA, *see* object oriented analysis
OOD, *see* object oriented design
OOP, *see* object oriented programming
OPEN statement, 74, 92, 97, 159, 271
operator, 27
.dot., 258
.op., 86, 165
.solve., 89, 90
.t., 166
.x., 166
assignment, 39
binary, 86
defined, 18, 86
extended, 86
overloaded, 18, 143, 149, 189
overloading, 39, 85, 258
symbol, 86
unary, 86
user defined, 76, 165
operator overloading, 10, 189, 260, 290
operator precedence, 52
operator symbol, 165
optional argument, 29, 37, 75
OPTIONAL attribute, 29, 36, 104, 137
OR operand, 37
OR operator, 53
order vector, 99
ordering array, 95
orthonormal basis, 256, 257
outer loop, 61
overflow, 290
overloaded member, 121
overloading, 39, 48, 85, 189, 290
operators, 42
testing, 86

package, 15
parallel computer, 43
PARAMETER attribute, 25, 29, 37, 60, 69, 70, 75, 82, 104, 112
Part-Of, 107
partial derivative, 176
partial differential equation, 183
partitioned matrix, 171
pass by reference, 57, 76, 87, 253
pass by value, 57, 58, 76, 253
pass-by-value, 290
path name, 37
pi, 25
Platypus, 194
pointer, 10, 23, 75, 86, 290
address, 150
allocatable, 15
allocate, 142
arithmetic, 87
array, 135
assignment, 88
association, 87
deallocate, 142
declaration, 87
dereference, 58
detrimental effect, 87
in expression, 88
inquiry, 88
nullify, 88
nullifying, 88
status, 15, 87
target, 87
writting, 150
pointer array, 290
pointer assignment, 290
pointer object, 131

- pointer variable, 86
- polymorphic class, 131
- polymorphic interface, 118
- polymorphism, 18, 33, 34, 119, 124, 194, 290
- pop, 137
- portability, 15
- pre-condition checking, 137
- pre-processor, 129
- precedence order, 53
- precedence rules, 11
- precision, 179, 192
  - double, 81
  - kind, 24
  - portable, 81
  - single, 81
  - specified, 81
  - underscore, 24
  - user defined, 24
- precision kind, 24
- PRESENT function, 29, 36, 37, 42, 75, 253
- PRINT * statement, 29
- private, 33, 104, 187, 290
- PRIVATE attribute, 29, 36
- private attributes, 37
- PRIVATE statement, 27
- procedural programming, 18
- procedure, 68
- PRODUCT function, 162
- product of array elements, 162
- program
  - documentation, 17
  - executable, 17
  - scope, 14
- program code
  - Another_Great_Arc, 270
  - array_indexing, 60
  - check_basis, 257
  - check_vector_class, 256
  - clip_an_array, 69
  - create_a_type, 26
  - create_Student, 37
  - Date_test, 37
  - declare_interface, 76
  - Dynamic_Dispatching, 131
  - Fibonacci, 29
  - game_of_life, 251
  - geometry, 34
  - if_else_logic, 63
  - linear_fit, 92
  - Logical_operators, 63
  - maximum, 70
  - Memory_Leak, 183
  - Memory_Leak_Counted, 274
  - Newton, 250
  - No_Copy_Reallocate, 183
  - operate_on_strings, 78
  - Person_inherit, 37
  - random_access_file, 151
  - Rational_test, 42
  - relational_operators, 63
  - Revise_employee_manager, 273
  - simple_loop, 60
  - string_to_numbers, 80
  - structure_components, 84
  - Testing_a_Queue, 142
  - Testing_a_Stack, 137
  - test_bubble, 97
  - Test_Conversion, 252
  - Test_doubly_linked, 149
  - test_Drill, 106
  - test_Employee_1, 122
  - test_four_classes, 121
  - test_Fractions, 86
  - test_Great_Arc, 112
  - test_inventory_system, 272
  - test_Manager_2, 123
  - test_Manager_3, 124, 133
  - Test_Physical, 252
  - test_singly_linked, 143
  - two_line_lsq_fit, 267
  - watch, 265
- program keyword, 56
- PROGRAM statement, 26, 29
- projectile, 101
- prototype, 6, 75
- pseudo-pointer, 95
- pseudo-random numbers, 162
- pseudocode, 5, 14, 51, 69, 101, 291
  - if, 13
  - if-else, 13
  - indexed loop, 9
  - nested if, 13
  - post-test loop, 9
  - pre-test loop, 9
- public, 33, 123, 136, 187, 291
- PUBLIC attribute, 29
- public constructor, 37
- public method, 27
- PUBLIC statement, 27
- push, 137

- quadratic equation, 3
- query, 191
- queue, 88, 135, 139

- raise to power, 56
- random access, 150

RANDOM_NUMBER subroutine, 162
RANDOM_SEED subroutine, 162
rank, 157, 291
rational number, 38, 39
read error, 102
READ statement, 29, 61, 75
real, 10, 81, 161
REAL function, 162
REAL type, 23, 24, 53
real whole number, 162
reallocate, 183, 195
recursive algorithm, 87
RECURSIVE qualifier, 42, 101
reference, 10
referencing components, 82
relational operator, 52, 53, 63, 77, 142, 143, 149
remainder, 56
rename, 119
rename modifier, 119
REPEAT function, 77
reshape, 158
reshape an array, 162
RESHAPE intrinsic, 162
RESULT option, 29
result value, 69
return, 157
RETURN statement, 65
REWIND statement, 75, 183, 265, 266, 268
round number, 56

sample data, 98
SCAN function, 77
scatter, 169
scope, 14, 291
SELECT CASE statement, 63, 188, 272
SELECTED_INT_KIND, 23, 24
SELECTED_REAL_KIND, 23, 24
selector symbol, 26, 29, 34
server, 18
SHAPE function, 162
short, 24
side effect, 142, 291
SIGN function, 162
signum, 162
SIN function, 56, 162, 249
sine, 56, 162
SINH function, 56, 162
size, 12
SIZE intrinsic, 69, 89, 92, 155, 162
smallest integer, 56
smallest number, 162
smallest positive number, 162
Smalltalk, 18
sort, 86, 90, 92, 95, 125
    bubble, 92
    characters, 94
    object, 96
    objects, 94
    strings, 94
sorting, 42
sparse matrix, 192
sparse storage, 263
sparse vector, 49, 149, 258
sparse vector class, 179
specification, 4, 190
SQRT function, 27, 56, 112, 162
square root, 27, 56, 68, 162
stack, 88, 135, 139, 291
STAT = variable, 74
statement, 2, 9
statement block, 12, 58
statements, 1
status
    FILE, 75
    IOSTAT=, 75
    MODE, 75
    OPENED=, 75
status checking, 157
STATUS= specifier, 271
stiffness matrix, 191, 192
STOP statement, 37, 70, 151, 181, 188
storage
    column wise, 155
    row wise, 155
string, 23, 56, 150
    adjust, 77
    case change, 80
    character number, 77
    collating sets, 77
    colon operator, 77
    concatenate, 77
    copy, 77
    dynamic length, 76
    from number, 80
    functions, 77
    length, 77
    logic, 77
    repeat, 77
    scan, 77
    to number, 80
    trim, 77
    verify, 77
strings, 76
strong typing, 53, 291
struct, 53
structure, 23, 25, 33, 84
structure constructor, 26

- structured programming, 13
- submatrix, 171
- subprogram, 68
  - recursive, 101
- subroutine, 68, 69
- subroutine code
  - Add_to_Q, 140, 142
  - allocate_type_application, 181
  - Alloc_Count_Int, 183
  - assign, 86, 131
  - assign_memb_1, 131
  - assign_memb_2, 131
  - Change, 76
  - deallocate_type_application, 181
  - Dealloc_Count_Int, 183
  - delete_Rational, 42
  - delete_Sparse_Vector, 258
  - delete_Vector, 255
  - destroy_D_L_List, 149
  - detroy_D_L_List, 149
  - display_all, 271
  - display_members, 131
  - display_memb_1, 131
  - display_memb_2, 131
  - D_L_insert_before, 149
  - enter_entry, 272
  - enter_item, 264
  - enter_update, 272
  - equal_Fraction, 86
  - equal_Integer, 42
  - equal_Real, 255
  - equal_Vector, 260
  - exception, 137, 140
  - exception_status, 75, 142
  - file_read, 264
  - file_write, 264
  - in, 104, 106
  - increase_Size, 271
  - initialize, 272
  - Init_Point, 188
  - Init_Point_Another, 188
  - Init_Point_Vctr, 188
  - Integer_Sort, 95, 97, 98
  - invert, 42
  - list, 42, 86, 255
  - List_Angle, 112
  - List_Great_Arc, 112
  - List_Position, 112
  - List_Position_Angle, 112
  - List_Pt_to_Pt, 112
  - list_type_alloc_status, 181
  - lsq_fit, 92
  - make_Sparse_Vector, 258

- mult_Fraction, 86
- MyPrint_Point, 188
- new, 131
- new_member_1, 131
- new_member_2, 131
- No_Change, 76
- nullify_Is_A_Member, 131
- orthonormal_basis, 257
- out, 104, 106
- pretty, 262
- Print, 29
- print, 121
- PrintPay, 123, 124
- PrintPayEmployee, 123, 124
- PrintPayManager, 123, 124
- print_Date, 37
- print_DOB, 37
- print_DOD, 37
- print_DOM, 37
- print_D_L_list, 149
- print_GPA, 37
- print_item, 264
- print_Name, 37
- print_Nationality, 37
- print_Sex, 37
- print_S_L_list, 143
- push_on_Stack, 137
- readData, 92, 100, 266
- read_Date, 37
- Read_Position_Angle, 112
- read_Vector, 255, 262
- read_xy_file, 268
- reduce, 42
- Remove_from_Q, 142
- Resize_Count_Int_OneD, 183
- restore_system, 271
- save_system, 271
- setData, 123
- setSalaried, 123, 124
- set_DOB, 37
- set_DOD, 37
- set_DOM, 37
- set_element, 262
- set_Latitude, 112
- set_Longitude, 112
- Set_Point, 188
- set_Size, 271
- Set_Vec, 188
- Set_X, 188
- Set_XY, 188
- show, 262
- show_Data, 97
- show_r_v, 262

  simple_arithmetic, 56
  Sort_Reals, 93
  Sort_String, 94
  Spy, 251
  String_Sort, 97, 98
  swap_objects, 126
  swap_real, 127
  swap_type, 128
  S_L_delete, 143
  S_L_insert, 143
  testing_basis, 257
  test_Manager_1, 123
  test_matrix, 89
  tic, 72
SUBROUTINE statement, 29
subroutines, 33
subscript, 26, 59, 155
  bounds, 155
  range, 177
  vector, 166
subscript triplet, 291
subtraction, 56
subtype, 131
subtyping, 124, 130
sum, 12
SUM function, 12, 69, 162
SUM intrinsic, 92, 165
sum of array elements, 162
super class, 119
syntactic error, 17
SYSTEM_CLOCK intrinsic, 72

tab, 78, 98, 102
TAN function, 56, 162
tangent, 56, 162
TANH function, 56, 162
TARGET, 15
target, 23, 75, 87, 88, 292
template, 43, 124, 126, 292
tensor, 155
testing, 15
time, 265
time of day, 99
TINY function, 162
Toeplitz matrix, 171
top-down, 4
total of elements in array, 162
transformational functions, 165
transpose, 159, 171, 173
TRANSPOSE intrinsic, 162, 166
tree, 292
tree structure, 38, 87, 88
tridiagonal matrix, 179
TRIM function, 77
triplet, *see* colon operator
true, 12
TRUE result, 62
truncate to real whole number, 162
truss, 166
type
  conversion, 80
  default, 52
  implicit, 52
TYPE declaration, 26, 29
TYPE statement, 27, 34

unary operator, 292
underflow, 292
unexpected result, 165
upper triangle, 171, 174
USE association, 119, 123, 190
USE statement, 29, 33, 34, 37, 85, 89
USE, ONLY, 119
user defined operator, 165
user interface, 2

validation, 29
variable, 8, 10, 23, 51
  global, 14
  name, 10
  type, 10
variable rank array, 156
vector, 155, 292
vector class, 48, 179, 252, 256
vector subscript, 61, 166, 292
VERIFY function, 77
volume, 48

weakness, 193
WHERE construct, 165
WHERE statement, 61, 66, 165
while-true, 67
wildcard, 126
WRITE statement, 34, 61, 75