

# Index

- abstract data type, 227–88
- acyclic digraph, 328
- ADA, user-defined ADTs, 258
- ADT
  - abstract data type, 227–88
  - binary tree, 237
  - queue, 228, 233
  - stack, 228
  - tree, 228, 237
- algebra, of diagrams, 46
- algorithm language, 77
- alternation, proof, 178–83
- and*
  - equational specification, 264–9
  - logical operator, 12, 157
- arc, of graph, 55
- assembly language, templates, 126–43, 143–54
- assertion, in program, 312
- assignment, 90, 100, 115, 129, 144
- assignment statement, 73
  - parallel, 355
  - PDL, 79
  - semantics, 217
- associative recursion, 359
  
- base step, of inductive proof, 198
- binary tree, 56
  - ADT, 237
- Boolean, type, 18
- Boolean algebra, 157
- BOOLEAN type, equational specification, 262
- built-in function, of PDL, 82
  
- call-by-name, 225
- call-by-reference, 225
- call-by-value, 224
- Cartesian product, 62
- case constructs, semantics, 222
- Char, type, 18
- CLEAR, specification language, 6
- COBOL, templates for, 115–26
  
- code
  - final, 72
  - target, 77
- code generation, 88–154
- coding template, 88
- complete tree, 56
- completeness, of ADT specification, 266
- compound data structure, 95, 107, 139, 151
- concatenation
  - equational specification, 270–1
  - of lists, 26
- consistency, of ADT specification, 266
- constant, in equational specification, 262
- constructed data type, 18
- control-flow diagram, 166
- control structures
  - assembly language templates, 129–38, 144–51
  - COBOL templates, 151–21
  - FORTRAN templates, 100–6
  - Pascal templates, 90–4
- correctness, of program, 322
- correctness theorem, 174
- cutpoint, in program analysis, 319
- cycle, in a digraph, 55, 327
  
- data, 11
- data structure
  - external, 98
  - file, 98
  - internal, 95
  - List, 24
  - Pair, 18
  - Set, 28
  - Triple, 21
  - Tuple, 22
- data structure diagram, 39
- data structures
  - assembly language templates, 139–43, 151–4
  - COBOL templates, 121–6
  - FORTRAN templates, 106–15
  - Pascal templates, 95–9

368 *Index*

- data type
  - constructed, 18
  - encapsulation, 258
  - in specification, 344
- data-flow diagram, 166, 331
- decision node, 302
- declarative programming language, 70, 72
- design structure diagram, 50
- diagram, 39
  - control-flow, 166
  - data structure, 39
  - data-flow, 166, 331
  - design structure, 50
  - fall-back notation, 50
  - Nassi–Schneiderman, 34
  - program structure, 39
  - repetition on, 39
  - selection on, 39
  - sequencing on, 39
  - syntax, 39
- digraph
  - acyclic, 328
  - cycle, 55, 327
  - directed graph, 55, 328
  - restricted to network, 55
- direct proof, 160
- directed graph, digraph, 55, 328
- DO-loop, semantics, 222
- documentation, 88
- does not imply, logical operator, 162
- domain, of an operation, 174
- domain rule, 174
- double recursion, 358
- dynamic storage allocation, 88
  
- edge, of graph, 55
- effect, of an operation, 287
- element
  - of a list, 24
  - of a set, 28
- embedded single recursion, 361
- empty list, 25
- empty set, 29
- encapsulation, of data type, 258
- enrichment, of an equational specification, 273
- enumerated type, 107, 121, 139, 151
  - in PDL, 80
- eqns**, components of equational specification, 264
- equals-for-equals, substitution, 264
- equational specification, 262–88
- err eqns**, error equations, in an equational specification, 271
- existential quantifier, 195
  
- expression
  - syntax in PDL, 81
  - syntax in specification, 346
- extended recursion, 362
- external data structures, templates for, 98, 110, 124, 143, 154
  
- fall-back, shown on diagrams, 50
- False, logical value, 15
- false acceptance, 177
- file
  - data structure, 98
  - random access, 98
  - sequential, 98
- flag
  - restructuring, 304
  - set by machine instruction, 127–9
- Flex/Algol68, user-defined ADTs, 260
- flowchart, 39
- flowgraph, 291
- folding, transformation step, 65
- for-all, logical operator, 195
- for-loop, semantics, 221
- FORTRAN
  - multiple-entry subroutines, 257
  - templates for, 99–114
  - user-defined ADTs, 256
- function, syntax in PDL, 78
- function definition, 93, 105, 120, 134, 149
- functional programming language, 72
- functional specification, 32
  
- ghost variable, 313, 321
- graph, 55
  - arc of, 55
  - directed, 55, 328
  - edge of, 55
  - node of, 55
  
- head*
  - of list, 25
  - of list, equational specification, 270–1
  
- identity process, 46
- if statement, 91, 101, 117, 131, 146
- if ... then ... else
  - formal definition, 164
  - proof, 179
- imperative programming language, 70, 72
- implication, logical operator, 156, 158
- induction, proof method, 195–9
- induction hypothesis, 198
- induction step, of a proof, 198
- initialisation of state, 336
- input, 12, 37
- input list, in specification, 98

- Integer  
 equational specification, 274–80  
 type, 18  
 internal data structures, templates for, 95,  
 107, 122, 140, 151  
 intersect  
 equational specification, 282–5  
 set operation, 28  
 invar, *see*: iterative proof, 206, 212  
 invertible recursion, 355  
 iteration, proof of iterative construct, 201–15
- label node, 302  
 layout, of PDL, 79  
 leaf, of tree, 56  
 linear ordering, 326  
 linear recursion, 360  
 linked tree, 290  
 list  
 concatenation, 26  
 data structure, 24  
 element, 24  
 empty, 25  
*head*, 25  
*head*, equational specification, 270–1  
 member, 24  
 of fixed length, 95, 107, 122, 140, 151  
 of variable length, 96, 108, 122, 140  
 restricted tree, 57  
*tail*, 25  
*tail*, equational specification, 270–1  
 LIST type, equational specification, 270  
 load, operation parameters, 167–8  
 logic programming language, 72  
 logical operator  
*and*, 12, 157  
 does not imply, 162  
 for-all, 195  
 implication, 156, 158  
*not*, 157  
*or*, 157  
 there-exists, 195  
 loop invariant, *see also*: invar, 319
- member  
 of list, 24  
 of set, 28  
 method of inductive assertions, 313  
*modus ponens*, 160  
 multiple recursion, 362  
 multiple-entry subroutines, FORTRAN, 257
- Nassi–Schneiderman diagram, 54  
 natural number, 18  
 network  
 restricted diagram, 55  
 restricted to tree, 56
- new, procedure for space allocation, 96, 109  
 node, of graph, 55  
 non-recursive implementation, in PDL, 242  
 non-result, in an equational specification,  
 268  
 non-terminal node, of tree, 56  
*not*  
 equational specification, 264–9  
 logical operator, 157
- OBJ specification language, 6  
 object form, of program, 289  
 operation, 13  
 domain of, 174  
 post-condition, 13  
 pre-condition, 13  
 range of, 174  
 sequencing of, 16  
 set intersect, 28  
 set membership, 28  
 set union, 28  
 set without, 29  
 state associated with, 33  
 subset of set, 28  
 type of, 13  
 operation specification, 345  
 operators, in PDL, 82  
 ops, component of equational specification,  
 264  
*or*  
 equational specification, 264–9  
 logical operator, 157  
 output, 12, 37  
 output list, in specification, 98
- package, in ADA, 258  
 Pair, data structure, 18  
 parallel assignment, definition, 355  
 parameter passing, 224–6  
 partial correctness, of program, 207, 322  
 Pascal, templates for, 88–99  
 PDL, 72–87  
 built-in function, 82  
 description of, 78–82, 348–52  
 enumerated type, 80  
 expression syntax, 81  
 layout, 79  
 list data type, 82  
 non-recursive implementation in, 242  
 operators in, 82  
 primitive type, 80  
 record data type, 80, 82  
 recursion removal, 242  
 recursive implementation in, 240  
 selector function, 80  
 statement syntax, 79  
 subrange type, 80

370 *Index*

- syntax, 80
  - type syntax, 80
  - union data type, 84
  - variable syntax, 81
- peek*
  - equational specification, 285–6
  - stack ADT operation, 229
- pop*
  - equational specification, 285–6
  - stack ADT operation, 229
- positive integer, 18
- post-check loop
  - diagram, 41
  - see also: repeat ... until*
- post-condition, of operation, 13
- postfix, operator notation, 228
- powerset, 63
- pre-check loop
  - diagram, 41
  - proof, 201
- pre-condition, of operation, 13
- pred*, predecessor function, 273
- predicate, 11, 156
- prefix, operator notation, 228
- primitive data type, 95, 121, 139, 151
- primitive type, in PDL, 80
- procedural specification, 32
- procedure call, 90, 100, 116, 130, 145
- process node, 302
- Program Design Language, PDL, 72
- Program Development Language, PDL, 72
- program structure diagram, 39
- programming language
  - declarative, 70, 72
  - functional, 72
  - imperative, 70, 72
  - logic, 72
  - semantics, 217
- proof
  - of alternation, 178
  - of *if ... then ... else*, 179
  - of pre-check loop, 201
  - of sequencing, 171
  - of transformation, 61
- proof method
  - direct, 160
  - induction, 195
- push*
  - equational specification, 285–6
  - stack ADT operation, 229
- quantifier
  - existential, 195
  - universal, 195
- queue, ADT, 228, 233
- random access file, 98
- range of an operation, 174
- range rule, 174
- Real, type, 18
- record, data type, in PDL, 80
- record, 108, 122, 142, 153
- recursion, 27
  - associative, 359
  - double, 358
  - embedded, single, 361
  - extended, 362
  - invertible, 355
  - linear, 360
  - multiple, 362
  - right-permutative, 356
  - tail, 352
- recursion removal, 353–64
  - from PDL, 242
- recursive implementation in PDL, 240
- refinement, of specifications, 32
- repeat statement*, 92, 103, 119, 133
- repeat ... until*
  - semantics, 220
  - transformation, 301, 307
- repetition, shown on diagrams, 39
- restructuring, of programs, 302–11
- results, 11
- right-permutative recursion, 356
- root
  - of tree ADT, 238
  - of tree, 56
- selection, shown on diagrams, 39
- selector function, in PDL, 80
- semantics
  - DO-loop, 222
  - of assignment statement, 217
  - of *case* construct, 222
  - of *for*-loop, 221
  - of programming language, 217
  - of *repeat ... until*, 220
- sequencing
  - of operations, 16
  - proof, 171–8
  - shown on diagrams, 39
- sequential file, 98
- set
  - data structure, 28
  - empty, 29
  - intersect operation, 28
  - membership test, 28
  - subset operation, 28
  - union operation, 28
  - without operation, 29
- set membership, equational specification, 282–5
- set type, equational specification, 282–5

- side-effect, of an operation, 287
- simplification, of expression, 65
- simply linked tree, 290
- size, of a set member, 196
- sorts**, component of equational specification, 264
- source form, of program, 289
- spaghetti code, 289
- specification, 14
  - data type, 344
  - equational, 262–88
  - functional, 32
  - input list, 98
  - of operation, 345
  - output list, 98
  - procedural, 32
  - refinement, 32
  - syntax, 344
  - transformation, 32, 61, 331
  - two level scheme for recursion, 202
- specification language
  - CLEAR, 6
  - OBJ, 6
  - VDM, 6
  - Z, 6
- stack
  - ADT, 228
  - peek* operation, 229
  - pop* operation, 229
  - push* operation, 229
- state
  - associated with an operation, 33
  - initialisation of, 336
- statement, syntax in PDL, 79
- statement sequence, 90, 101, 116, 131
- subrange type, 107, 121, 139
  - in PDL, 80
- subset
  - equational specification, 283–4
  - operation, 28
- substitution
  - equals-for-equals, 264
  - transformation step, 65
- subtree, of tree ADT, 238
- succ*, successor function, 272
- syntax, of PDL, 78, 80
- syntax diagram, 39
- tail*
  - of list, 25
  - of list, equational specification, 270–1
- tail recursion, 195, 335, 354
- target code, 77, 217–26
- template code, 88
- templates, 217–26
  - for assembly language, 126–43, 143–54
  - for COBOL, 115–26
  - for FORTRAN, 99–114
  - for Pascal, 88–99
- term, *see*: iteration proof, 207, 212
- terminal node, of tree, 56
- there-exists, logical operator, 195
- three-valued logic, 165
- to-end, *see*: iteration proof, 207, 212
- topological sort, 326
- total correctness, of program, 322
- transformation
  - folding step, 65
  - of specification, 32, 61, 331
  - proof of, 61
  - substitution step, 65
  - unfold–fold, 65
  - unfolding step, 65
  - while* loop, 191
- transitive property, 159
  - equational specification, 276
- tree
  - ADT, 228, 237
  - binary, 56
  - complete, 56
  - leaf, 56
  - linked, 290
  - non-terminal node, 56
  - restricted network, 56
  - restricted to list, 57
  - root, 56, 238
  - simply linked, 290
  - subtree of, 238
  - terminal node, 56
- Triple, data structure, 21
- True, logical value, 15
- truth value, = logical value, 15
- Tuple
  - data structure, 22
  - notation for, 62
- two level scheme, for recursive specification, 202
- type
  - abstract data type, 227
  - Boolean, 18
  - BOOLEAN, equational specification, 262
  - Char, 18
  - constructed data type, 18
  - equational specification, 264
  - Integer, 18
  - INTEGER, equational specification, 274–80
  - LIST, equational specification, 270
  - of an operation, 13
  - PDL list data type, 80, 82
  - PDL record data type, 80, 84
  - PDL union data type, 84
  - primitive in PDL, 80

372 *Index*

- Real, 18
  - set, equational specification, 282–5
  - syntax in PDL, 80
- undefined, logical value, 165
- unfold-fold transformation, 65
- unfolding, transformation step, 65
- union
  - data type in PDL, 84
  - equational specification, 282–5
  - set operation, 28
- universal quantifier, 195
- until . . . do*, transformation, 300
- update, of state, 167–8
- variable, syntax in PDL, 81
- VDM, specification language, 6
- verification, 155–216
- well-ordered set, 200
- well-ordering, 322
  - see also*: size, 200
  - while* loop, transformation, 191
  - while* statement, 91, 103, 118, 133
  - while . . . do*, transformation, 307
- without
  - equational specification, 283–5
  - set operation, 29
- Z, specification language, 6