

INDEX

Special Characters

"" (stringification) pseudo-operator, 467
"" key, 501
\$" special variable, 261, 306
\$\$ special variable, 155
\$. special variable, 353
\$/ special variable, 261, 361
\$; special variable, 84
\$;\$ prototype, 338
\$@ special variable, 430
\$_ special variable, 142–43, 160
& notation in Linogram, 495
&@ prototype, 337
&@ prototype, 338
&& @rest condition, 211
(?!) regex lookahead operator, 363
* operator, 273, 389, 558
* quantifier, 436
* token, 376
** operator, 426
+ operator, 558
+ quantifier, 436
+ token, 376
-> operator, 176
. operator, 176
; symbol in prototypes, 179
<...> operator, 187, 359, 360–65
 as rudimentary parser, 359
<=> operator, 102, 108
== operator, 168, 177–78
 for comparing objects, 477
? quantifier, 436
@_, modification of, 91–92
[...] operator, 167
\ (backslash) operator, 84–85, 87, 179, 439, 449
\@\$\$ prototype, 293
\@\@ prototype, 178
| operator, 437
||= operator, 106, 106–7
|| operator, 103

A

-a command-line option, 149
\$a and \$b variables, 345
Abelson, Harold, xvii
absolute zero, 475
abstract base class, 25, 478
abstract syntax tree (AST), 57–58, 393, 400, 408, 437, 553–54, 557
abstraction of functionality, 16, 41, 225, 412
accessor method, 29, 477
actions, 44, 49–50, 53, 446
Adler, David, 145
agenda, 207–12, 365, 370, 380–81
agenda method, universality of, 217
aggregation, 19–20
airplane, 109
algorithm, 10, 11, 287, 471
aliasing, 70–71, 81, 92, 112
alternate universe, 116–17
alternation of parsers, 387–88
ambiguity in arithmetic expressions, 54
amortization, 310–13
anchor component of URL, 189, 199
anonymous array, 31, 50, 81
anonymous functions, 19, 76, 79, 86, 123, 258–59, 300, 327, 371, 398, 421, 453, 478
 debugger's treatment of, 415–16
argument isn't numeric warning, 108
argument normalizer, inlined cache manager with, 90–92
arithmetic expressions, 54, 376, 390–435
 overview, 390–400
 calculator, 400, 424–27
 debugging, 415–24
 generic-operator parsers, 412–15
 grammar for, 376
 left recursive, 400–8
arrays, 66, 179, 181
 contrasted with iterators, 117–18

 contrasted with linked lists, 257
 looping over, 177–82
 representation of database rows as, 165–66
 repeated copying of, 129–30
arrow, definition in Linogram, 498
Ashton, Elaine, 145
assembly language, 314
associativity of operators, 257, 401–2, 426
AST (abstract syntax tree), 57–58, 393, 400, 408, 437, 553–54, 557
asymmetry, in Linogram parameter specifications, 559
atomic expression, 436
atomic features in Linogram, 494
atomic ray, 185
atoms in regexes, 436, 439
attribute–value pairs in HTML tags, 26
automatic profiling, 111–12
auxiliary parameter, 240

B

\b regex metacharacter, 272, 439
backslash character, 84–85, 87, 179, 439, 449
backtracking, 207, 456–65
 in regex engine, 364
backtracking parsers, 456–65
 overview, 456
 continuations, 457–61
 parse streams, 461–65
backup program, 206
bacteriophage, 135
bad link detector, 191, 192, 194
bad luck, 168–69
balanced parentheses, 287–88
barbarian, 206–7
base case, 3, 5, 8, 35, 211, 534
bear, 359–60

- BEGIN block, 338–39, 346, 350–51, 353, 355–56, 502
- beginner mode via alternative dispatch table, 49
- behavior function in constraint system, 479, 480
- Bell Labs, 563
- BFS (breadth-first search), 189, 200, 213–15
- binary, 1–3, 239–41
- binary search, 292–93
- binding, 71–72
- bio-informatics, 135–39
- Bird, Richard, xvii
- blessing, 356, 416, 466, 468–69
- blocks, 72–73, 390
 - bare, 67–68
 - as arguments to `sort()` function, 102
 - BEGIN, *see* BEGIN block
 - disk, *see* disk blocks
 - labelled, parser for, 541–42
- boolean “not” operator, 354
- Borges, Jorge Luis, 417
- Boss, the, 80
- box, definition in Linogram, 497
- breadth-first search (BFS), 189, 200, 213–15
- browsers, 68
- buffer, 363
- bullet symbol, 441
- Bunce, Tim, 164
- Burke, Sean, xvii–xviii, 29, 35
- bytes, 19–20
- C**
- C programming language, xv–xvi, 67, 75, 79–80, 105, 108–9, 153, 166, 344, 362, 365, 424, 501
 - auto variables in, 6
 - for loop, 173
 - library of, 153, 155
- C++ programming language, 25
- /c regex modifier flag, 367
- cache data, expiration of, 97–98
- cache hit, 65, 90, 104, 107
 - false, 84, 93, 107
- cache management overhead, 65, 87, 101, 109
- cache memory, 68, 108
- cache miss, 65, 91, 95, 107
- caching and memoization, 63, 65, 83, 90, 101, 268
 - alternatives to, 101–8
 - benefits of speed, 109–13
 - caveats, 80–84
 - evangelism, 108–9
 - inline caching, 66–68
 - in object methods, 96–99
 - key generation, 84–95
 - Memoize module, 70–80
 - persistent caches, 100–1
 - quantitative analysis of, 65, 83–84, 90, 101
 - and recursion, 65–66
- calculator, 54–61, 365–69, 400, 424–27, 435, 469
- calculus, 302, 305, 314, 319
- call frame, 231–32
- call tree, 215–17, 220
- callbacks, 10, 20, 23, 119, 127, 143–44, 158, 160, 181, 187–88, 190, 225, 227, 326, 371, 386
- caller() function, 420
- canonical form, 218–19
- canonical order, 349–50
- cantankerous grandfather, 228
- cardioid, 490
- Carp module, 186, 522, 532, 558
- carriage return, 439
- carrot, 162, 264, 275
- CGI, 189, 197
- CGI::Push module, 155
- chained lexers, 368–74
- character classes, 273, 281
- CHDIR directive in configuration file, 42
- chdir() function, 42
- Christiansen, Tom, xv, 129
- chromosome, 135
- chronological order, 102–6, 149, 299–300
- CIA, 184–85
- circle, definition in Linogram, 499
- clauses in grammars, 376
- cleverness, 211
- clock, memoized, 82–83
- clone method in URI::URL, 199
- closedir() function, 15
- closure, 76, 124, 326
- closure property, 76–78
- closure operator (*), 273, 286
- cmp operator, 105
- CMYK, 63–64
- code reference, 9–10, 41, 88, 159, *see also* closure, anonymous function
- code value (CV), 75–77
- coefficient, 317, 500, 508
- college student, penurious, 206
- comma-separated list, 542, 555
- comparator function, 101–2, 291–92, 350
- compiler, 57
- comp.lang.perl.misc newsgroup, 272
- components, 27
- compound interest, 310–13
- compromise, 85
- compulsive behavior, 453
- concatenate() operator, 397–98
- concatenation, 3, 273, 386–87
 - of parsers, 386–87
 - of regexes, 273, 275–79, 283–84, 320–21, 438
- configuration table, 41–42, 47
- CONSTANT object, 514–16
- constant part of equation, 500–1, 503
- constants, 483
- constraint networks, 487
- constraint systems, 472
- Constraint class, 512–13
- constraints section in Linogram specification, 549–50
- Constraint_Set class, 513–14
- _content key in HTML::TreeBuilder, 27
- context, 104, 118–19, 159, 165, 182–83, 192, 193
 - Linogram evaluation environment, 557–60
- continuation, 457–61
- contradiction, 509–13
- control-backslash character, 84
- conversion, decimal to binary, 1–3
- core dump, 75
- cos() function, 313–14, 319
- cosh() function, 320
- cost-benefit ratio, 110
- counting process, 131–32
- CPAN, 104–5, 152, 324, 500
- CPU, 68
- crab cakes, 82
- current matching position of scalar, 366–67
- Curry, Haskell B., 327
- currying, 326–31, 340–43, 346, 350, 353, 484

- custom key generation, 94–95
 - cutsorting, 288–300
 - log files, 293–300
 - cutting function, 290–91
 - CV (code value), 75–77
 - `cwd()` function, 46
 - Cwd module, 46
- D**
- d operator, 124
 - dangling symbolic link, 19, 23–24, 183, 325
 - database handle, 141
 - database query languages, 471
 - database-query parsing, 448–56
 - overview, 448
 - lexer, 448–51
 - parser, 451–56
 - databases, 351–57, 448–56
 - of cached values, 100–1, 111
 - flat, 140–53
 - SQL, 163–68
 - `Data::Dumper` module, 392
 - data structure, 84, 542
 - date format, ISO, 105
 - `DB_File` module, 100
 - DBI module, 163, 165–66, 168, 173
 - DBM, 101, 349
 - dead ends, 189
 - debugger, 154–55
 - debugging, 154–55, 415–17, 429, 475, 479, 551
 - decimal to binary conversion, 1–3
 - declarative programming, 471–563
 - overview, 471–72
 - linear equations, 488–90
 - local propagation networks, 472–88
 - overview, 472–75
 - implementing, 475–87
 - problems with, 487–88
 - declarator in Linogram, 545, 548
 - default action for dispatch table, 52–54
 - Defenestration of Prague, 204
 - defined** operator, 167
 - depth-first search, *see* DFS
 - derivation of sentence, 377
 - derivative function, 305–6, 319, 320, 331
 - DESTROY method, 97
 - destructive function, 91, 262, 295
 - destructive method, 506, 511
 - DFS, 189, 200, 203, 212–15, 228–29, 383, 403
 - ambiguity of, 228
 - bound on size of agenda, 213
 - contraindications for, 213–14, 379, 403
 - recursion and, 126, 189, 203, 214, 217
 - web spiders and, 214, 379
 - diagnostic messages, 13, 42, 201, 260–61, 426, 476
 - diamond**, definition in Linogram, 499
 - `die()`, throwing exceptions with, 430
 - differential calculus, 305, 319
 - Dijkstra, Edsger Wybe, 219
 - directives, 45–46
 - directory tree, 123, 161
 - directory walking, applications and variations of, 16–25
 - `dirhandles`, 13–15, 118
 - disk blocks, 15
 - diskettes, 206
 - dispatch tables, 41–61, 182, 438, 515–16, 520, 528, 551, 559
 - overview, 41
 - advantages of, 45–49
 - calculator, 54–61
 - configuration file handling, 41–54
 - default actions, 52–54
 - division as inverse of multiplication, 482–83, 515–16
 - division by zero, 432, 481–82, 517
 - division of power series, 320–24
 - division of treasure, *see* partition problem
 - do-it-yourself, 126
 - do-while** loop, 243
 - domain name server, 68, 108
 - DONGS HLAGHAGHL, 386–87
 - doorbell, 386–87
 - `drand48()` function, 153
 - draw** section in Linogram specification, 550–51, 553–54
 - drawables, 536–39, 550–51, 553–54
 - drawables list, 554
 - drawing function, 494–95, 537, 561
 - drunken fraternity brothers, 156
 - du** command in Unix, 16
 - duration (of value), 73–76
- E**
- e command line option, 149
 - e operator, 19
 - `each()` operator, 182, 349
 - `$elementfunc` argument, 29, 59–61
 - email, 293–94, 363–64
 - address, 53–54, 294
 - header of, 363
 - empty list, 24–25, 36, 166, 179, 346–47, 385
 - empty string, 273, 285, 372, 438, 511
 - end tag in HTML, 26
 - Engineering Mathematics Handbook*, 324
 - entropy, 156
 - environment, 76
 - `Environment` class, 538–39
 - epoch format, 295
 - `eq` operator, 169, 452
 - `Equation` class, 500–11
 - `Equation::System` class, 508–12
 - equations,
 - for Fibonacci numbers, 34
 - inconsistent, 509–10
 - linear, 488–513
 - redundant, 508
 - roots of, 301
 - solving, 301, 489
 - trivial (0=0), 502
 - equivalence classes, 218–19
 - error diagnosis and recovery in parsing, 427–35
 - escape sequence, 85
 - eta-conversion, 389–90, 391–92, 423, 437
 - Euclid's algorithm, 231
 - `eval` operator, 46, 86–88, 339, 340, 430, 543
 - exaggeration, 35
 - exceptions, 430–32
 - propagation of, 432
 - `execute` method in DBI module, 168
 - EXHAUSTED operator, 172–73
 - exhaustion, 122, 163, 189, 207, 209, 292, 350, 365, 370, 428, 459
 - explicit function for, 171–73
 - synthetic representation of, 167–73
 - `exists()` operator, 107
 - `exp()` function, 320
 - expiration of cache data, 97–98, 101
 - explicit exhaustion function, 171–73
 - explicit specification of empty list value, 346
 - explicit stacks, 242–53
 - exponentiation operation, 425
 - `Expression` class, 554

expression parser, 55–56, 376, 381–84, 391–415
 expressions, 376–77
 in Linogram, 554–58

F

-f operator, 13
 @F special variable, 149
 façade pattern, 170, 172–73
 factor of arithmetic expression, 394
 factorial function, 3–6, 241, 314–15, 324
 fame and renown, 415
 family tree, 489
 Fcntl module, 142
 features in Linogram, 490, 491–98, 520–21, 528–37
 Fenchurch, 82
 FETCH method, 184, 184–85
 fetchrow_arrayref method in DBI module, 165, 168, 173, 174
 Fibonacci numbers, 33–35, 65, 67, 243–53
 eliminating recursion from calculation of, 243–53
 fields in email message header, 363
 fields in flat text database, 140, 148, 150
 File::Basename module, 191
 File::Find module, 126
 File::ReadBackwards module, 152
 file system, 12, 26
 filehandle, 115–16, 186
 as iterator, 115–17
 iterator interface to, 139–40
 reading in while loop, 187
 tied, 186–87, 299
 files, 12–13, 18, 23–24, 82, 127, 142
 configuration, 41–52
 for persistent caching, 100–1
 GIF, 64
 INI, 117
 log, 148–53, 294–95
 with octopuses, 161
 with plutonium, 116–17
 filters and transforms, 157–63
 fishing expedition, 109
 flat databases, 148, 452
 FlatDB module, 140–53, 356, 448
 FlatDB::Iterator module, 150–52
 flat-file database, 140, 140–48
 floating-point numbers, 205–6
 flow chart, 157

foolishness, 80, 88, 287
 forcing the promise, 258, 269
 for loop, C-style, 173
 foreach, using to loop over more than one array, 177–82
 fork() function, 80
 Fortran, 6
 four-dimensional beings, 518
 fractions, 231
 fragment of URL, *see* anchor component
 FreezeThaw module, 85
 Frege, Gottlob, 327
 Frequently Asked Question, 1, 16, 129
 fruit, 234
 function calls, implementation of, 229
 function factory, 70–71, 199, 259, 325, 385, 406–7
 functional vs. object-oriented programming, 25–26
 functions,
 custom key generation, for impure functions, 94–95
 higher-order, 325, 333
 that return references, 81–82
 very fast functions, 83–84
 whose return values do not depend on their arguments, 80
 with reference arguments, 93
 with side effects, 80
 Funes the Memorious, 417

G

\G regex metacharacter, 366–67
 /g regex modifier flag, 366
 garbage collection, xv, 15, 73, 77, 79
 Gaussian elimination, 500, 502–12
 GCD (Greatest Common Divisor), 231
 geeks, 310
 General Electric, 69
 generic-operator parsers, 412–15
 Genomic sequence generator, 135–39
 getc() function, 186
 get_links method of
 HTML::LinkExtor module, 189–90
 GIF file, 63–64
 globs, 71, 186, 330, 332
 glob() operator, 119
 global filehandle, 48
 global variables, 5, 47, 49, 142–43, 264, 345, 420–21, 454

 effect on recursion, 5–6, 13–14
 filehandles as, 13–14, 15, 46
 hashes instead of, 51
 random number generator as, 156
 Goff, Jeff, 217
 golden ratio, 35
 golden_rectangle, definition in Linogram, 499
 goto, 232, 243, 466
 grammar, 376–80
 Grasshopper, 188–201
 greatest common divisor (GCD), 231–33
 grep() function, 333–35
 grocery bags, 35
 Guttman, Uri, 152

H

hack, 84
 hairy code, 414
 Hall, Joseph, 106
 Hamming problem, 269–72
 Hamming, Richard W., 269
 hash, 20, 23, 27, 39, 51, 66–67, 83, 89, 112, 200, 235, 296, 479, 494–95, 512, 518, 561
 cache, 64–67, 69, 71, 96, 101
 as dispatch table, 44
 for canonicalizing value, 218
 iterator component of, 119, 182
 power set of, 235–36
 tied, 100, 416
 hash key, 84, 93, 97
 hash slice, 144
 hash value computation, 83
 head of linked list, 256, 375
 header of email message, 363
 header of flat-file database, 140
 headers of HTML document, 30
 hexadecimal, 96
 hierarchical data, 12–15, 128, 203, 530
 Hietaniemi, Jarkko, 234–35
 high school, 300, 312, 489
 higher-order functions, 325, 333
 currying, 325–57
 reduce() and combine(), 343–51
 hline, definition in Linogram, 496
 hoax, 69
 Hoefler, Jonathan, 204
 Hoefler Type Foundry, 204
 hook, 171, 188, 201, *see also* callback

HTML, 26–33, 59–61, 179, 188–89, 326–27
 HTML::LinkExtor module, 188, 190
 HTML::TreeBuilder module, 27, 27–29
 HTTP, 150
 httpd log file, 148–49
 Human Genome Project, 135
 hyperlinks, 188–97, 200, 214

I

`if-else` tree, 42–44, 56, 59, 245
 impure function, 94–95
 INCLUDE directive in configuration file, 48
 incrementing numerals, 132
 India, 299
 infinite loop, 3, 117, 308, 403, 477, 506
 infinite sequence of integers, 260
 infinite streams, 255–324
 overview, 255
 hamming problem, 269–72
 lazy linked lists, 257–63
 Newton-Raphson method, 300–13
 power series, 313–24
 recursive streams, 263–69
 regex string generation, 272–300
 infix form of expression, 54
 ingenuity, 48, 68, 124–26, 134, 179, 207, 323
 INI file, 117
 inline caching, 66–68
 inlined cache manager, with argument normalizer, 90–92
 inlining, 86–88, 90–92, 106
 internal stack, 125
 intrinsic constraints, 520–22, 534–35
`Intrinsic_Constraint_Set` class, 521–22
Introduction to Functional Programming, xvii
 i-number, 294
 investment banking, 96–98, 106–7
 IOU, 259–60
 IRC, 16, 272
`is_in()` example, 93
`isa` method, 88
 ISO date format, 105
 iteration, C model of, 173–74
 iterators, 115–201
 overview, 115

alternative interfaces to, 177–87
 converting recursive functions to, 203–53
 examples, 126–57
 exhaustion of, 122
 filehandle iterators, 139–40
 filters and transforms of, 157–63
 flat-file database, 140–48
 genomic sequence generator, 135–39
 homemade, 119–26
 introduction, 115–19
 kicking, 121, 157, 160, 185, 195
 for permutations, 128–35
 random number generators, 153–57
 rewinding, 255
 searching databases backwards, 148–53
 semipredicate problem, 163–76
 web spiders, 187–201

J

jargon, 377
 Junko, 82

K

KEEPER label for HTML elements, 30–31, 326
 key generation, 84–95
 overview, 84–88
 functions with reference arguments, 93
 partitioning, 93–94
 key-generation method, 85
`keys()` function, 119
 kicking an iterator, 121, 157, 160, 185, 195
 Kleene closure, 273
 Knuth, Donald E., 154, 309

L

`-1` operator, 19
 labeled block, 541–42
 Latin, 80, 207
 laws of programming, 43
 lazy linked lists, 257–63
 left-associative operator, 426
 left recursion, 400–8, 403–4, 409
 left-recursive grammar rules, 403–4
 Leler, Wm, xviii
 lemniscate, 490
 Leonardo of Pisa (Fibonacci), 33

lexer, Perl's, 450–51
 lexers, 365–75, 441–42, 448–51, 540
 overview, 359–60
 chained, 368–74
 emulating `<>` operator, 360–65
 peeking, 374–75
 lexical analysis, *see* lexers
 lexical closure, 76
 lexical filehandle, 48
 lexical variables, 5–6, 14, 67, 73, 92, 418
 lexicality, 72–73, 79
 lexing, *see* lexers
`line`, definition in Linogram, 494
 linear equations, 488–90
 linked lists, 255–57
 head, 256
 lazy, 257–63
 splicing into, 257
 tail, 256
 linogram, 490–500
 overview, 490–500
 equations, 500–14
 feature types, 530–39
 missing features, 560–63
 parser, 539–60
 overview, 539–40
 declarations, 545–54
 definitions, 543–44
 expressions, 554–60
 required extensions, 541–42
 programs, 543
 scalar types, 531–32
 %TYPES, 542
 values, 514–30
 overview, 514–16
 constant values, 516–18
 feature values, 520–21
 feature-value methods, 527–30
 intrinsic constraints, 521–22
 synthetic constraints, 522–27
 tuple values, 518–20
 Lisp programming language, xv–xvi, 107
 list assignment as condition, 165–66
 list context, 104, 119, 159, 165, 182–83, 192, 193
 list expressions, 390
`List::Util` module, 343–47
 literal regexes, 274
`local()` operator, 142–43, 160, 306
 local propagation networks, 472–88
 overview, 472–75
 implementing, 475–87
 problems with, 487–88

log files, 148–53, 293–300, 348–49
log() function, 320
 LOGFILE directive in configuration file, 42
 lookahead assertions, 439
 lookbehind assertions, 439
 loop control operators, 243
 love and admiration, 17
ls command in Unix, 16
ls -R command in Unix, 16
 Lucas, Edouard, 6
 Luke the Hermit, 107
LWP::Simple module, 188–89, 201

M

-M operator, 82–83
m//g operator, 119
 Macdonald, John, 234
 Machol, Richard Morris, 228
 Macintosh, 206
 magic variable, 184, 186
mailto URL, 200
 maintenance programmers, 87
make_counter example, 77–79
malloc() function, 166
map() function, 124, 157–60, 262, 333–36, 345, 397, 521, 523
 Marshall, Edward Waite, 69
 marshalling, 69
 mask function, 170, 172–73
Mastering Algorithms with Perl, 234, 293
Math::BigFloat (multiprecision floating-point library), 307, 435
Math::BigRat module, 324
Math::Complex module, 435
 mathematics geeks, 310
 matrix transformations, 500
max() function in **Scalar::Util** module, 343–45
maxstr() function in **Scalar::Util** module, 344
 MAYBE label for HTML elements, 30–31, 326
 meaning of parsed input, 376, 384
 meiosis, 261
Memoize module, 69–80, 83, 105
 memoization, *see* caching and memoization
 memoized clock, 82–83
 merging streams, 270–71, 274–75
 Michie, Donald, 69
min() function, 343–44

minstr() function, 344
ML for the Working Programmer, xvii
 monster module, 70
 morphology, 359–60
 mortgage, 310–13
 Mozilla, 198
 MSIE, 198
 multiple-list iterator, 179
my declaration, 5, 13, 67, 72–73
 oddity in semantics of, 264
my variable, *see* lexical variables
 mystery, 185

N

\n (newline) character, 362–63
-n command-line option, 149
%N hash for naming parsers, 416–23, 467
 nasty regex, 287
 nasty surprises, 142, 229, 335, 427
 national security, 185
 network server, 68, 94–95, 108, 148, 189
 newline character, 261, 363, 441
 newsgroups, 16, 272
 Newton, Isaac, 301
 Newton-Raphson method, 300–13
nextval method, 176
 NEXTVAL operation, 115–16, 120–21, 171–73
 90-10 rule, 110
 nodes, 212–13, 215–17, 225–28
 in AST, 393, 400
 in **HTML::TreeBuilder** object, 27–29
 of linked list, 256–63
 in local propagation network, 472–74
 of notional tree of sentential forms, 377, 380, 382, 403
 in outline structure, 442–44
 non-reentrant, 6
 nonterminal symbols, 377
 normalization of equations, 511
 Norvig, Peter, xv, xvii
 null function, 24, 60
 null pointer, 166
 NULL value in SQL databases, 163, 165
 numerals, canonical form for, 219

O

O() notation, 103
 object constructor methods, 75, 81, 184, 187, 501–2, 516, 518, 527

object identity, testing with **==**, 477
 object methods, caching in, 96–99
 object-oriented (OO) programming, 25–26, 75, 96–98
 octopuses, 81–82, 127–28, 161
 oddity, 24, 264
 odometer method, 132–34, 138
 OO (object-oriented) programming, *see* object-oriented programming
open() function, 116
opendir() function, 13
 operating system, 68, 82–83, 149–50
 operator overloading, *see* overloading
 operator precedence, 357, 372, 393–94, 414–15, 437, 465
 optimization, 81, 110–11, 171, 204, 209–11, 230–31, 240–41, 253, 322, 332
 option checking, 201
 optional item, parser for, 541
 orcish maneuver, 106
 Orwant, Jon, 234
 out of scope, 15, 71–75, 92, 264
 outlines, 440–48
overload module, 357, 467
 overloading, 356, 416, 465–70
 oxcart, 109

P

pad, 73–77
Pair objects, 468
 palindromes, 214–15
 parabola, 301–3
Paradigms of Artificial Intelligence Programming, xv, xvii
param declaration in Linogram, 560–62
 parameter specifications in Linogram, 545–49, 552, 559–60
 parameters, 560–61
 parentheses,
 balanced, 287–88
 disambiguating expressions, 54, 58, 437
 in output of parser, 393–94, 398
 in regex, 364
parse method in **HTML::LinkExtor**, 190
 parser function, 551
 parser operator, 390
 parser, Perl's, 159, 360
Parser module, 384

- parsing, 43, 117–18, 198, 199, 351, 359–470
 - overview, 359
 - arithmetic expressions, 390–435
 - backtracking parsers, 456–65
 - database-query parsing, 448–56
 - HTML, 27, 188–90
 - in general, 376–84
 - lexers, 359–75
 - Linogram specifications, 539–57
 - overloaded operators for, 465–70
 - recursive-descent parsers, 384–90
 - regexes, 435–40
 - partial sums, 316
 - partition problem, 35–39, 93–94, 204–12
 - partitioning, 35–39, 93–94
 - partitions of an integer, 217–25, 225–26
 - comparing, 224
 - `pathfinder.com`, 198
 - patterns, 131, 135, *see also* regexes
 - Paulson, Lawrence, xvii
 - penurious college student, 206
 - performance, 63–64, 80, 93, 106, 110, 266–68, 332, 453
 - period of random-number generator, 154
 - Perl 6, 176
 - Perl interactive debugger, 415
 - `perldoc` program, 112
 - permutations, 3–4, 128–35
 - persistent caches, 100–1
 - pi (π), 317–19, 426
 - `pic` program, 563
 - pipe, 117
 - plain text, 26
 - plumber, 206–7
 - plutonium, 94–95, 116
 - `point`, definition in Linogram, 494
 - polymorphism, 331
 - `pop`, 126, 223
 - `POPDIR` directive in configuration file, 46–47
 - `pos()` function, 366–67
 - postfix form, 54
 - postprocessing function, 544
 - PostScript, 494, 496
 - power series, 313–24
 - overview, 313–19
 - derivatives, 319
 - symbolic computation, 320–24
 - power series expansion, 313–24
 - power set, 234
 - powers of 2, 265–69
 - Prague, Defenestration of, 204
 - precedence, 357, 437, 465
 - in arithmetic expressions, 393, 415
 - predicate function, 107, 160–62, 194, 263, 333, 537
 - pre-emption, 223
 - preprocessor, 139
 - primitive features in Linogram, 491, 531–32
 - `print()`, as expression, 424
 - `print` statement, 9, 416
 - printer, 63
 - `printf()` function, 19
 - priority queue, 225, 228
 - private variables, *see* lexical variables
 - probability, 65
 - productions in grammars, 376
 - profiler, 110–11
 - profiling, 110–12
 - Prolog programming language, 471
 - promise, 257–58
 - forcing, 258, 269
 - propagation network, 485–86
 - prototypes, 123, 159, 178, 179, 236, 293, 304, 334, 337–43, 346
 - proxy server, 108
 - pruning, 213, 380, 382, 403
 - pseudo-random, 153
 - pun, 122, 374
 - punctuation, excessive, 144
 - pure functions, 82
 - pure virtual methods, 25
 - `push()` function, 126
 - `PUSHDIR` directive in configuration file, 46
- ## Q
- `q{...}` syntax, 87
 - `qmail` mail system, 293–94, 299
 - quantification of regexes, 437
 - queue, 124–26, 127, 188, 192–93, 201
 - queue members, 193
 - `qw(...)` syntax, 150
- ## R
- rabbits, 33–34
 - `rand()` function, 153
 - `random()` function, 153
 - random number generation, 153–57
 - Raphson, Joseph, 301
 - `readdir()` function, 118
 - READLINE method, 187
 - reciprocal, 516–17
 - of power series, 323–24
 - recursion, 1
 - and caching, 65–66
 - as DFS, 215–17
 - elimination of, 229–53, 348
 - recursive-descent parsers, 384–90
 - overview, 384
 - compound operators, 388–90
 - parser operators, 386–88
 - very simple parsers, 384–86
 - recursive streams, 263–69, 280, 323
 - redo**, 243
 - re-entrancy, 6, 48–49
 - reentrant function, 49
 - `ref()` function, 88, 501–2
 - `ref($base) || $base` technique, 501–2
 - refactoring, 222–23, 234–39, 241, 249–53, 328
 - reference arguments, functions with, 93
 - references, 9–10, 19, 41, 61, 71, 73–76, 88, 164–65, 467, 502, 554
 - comparing with `==`, 168–69, 178
 - from pad, 75, 76
 - functions returning, 81–82
 - functions accepting, 93
 - stringification of, 88, 93, 169, 415–16, 467
 - referrer, fake, 193
 - referring URL, 192
 - regex engine, 119, 471
 - regex string generation, 272–300
 - regexes, 119, 143, 272, 286, 361, 365–66, 368–69, 435–40, 471
 - as examples of declarative programming, 471
 - as lexers, 365
 - syntax of, 437
 - relaxation, 488
 - repetition of parser operation, 389–90
 - return-value parameter, 240
 - `reverse()` function, 135
 - RGB, 63–64
 - robot policy, 197
 - `robots.txt` file, 197–98
 - Roode, Eric, xviii, 178, 178–79
 - root feature in linogram, 490
 - root of equation, 301
 - ROOT type in Linogram, 542
 - round-off error, 305–6, 307–8, 322
 - RPN (reverse Polish notation), 54–55

S

- s operator, 12, 15
- /s regex modifier flag, 450
- s/// operator, 87
- scalar, 9, 183–84, 366
 - tied, 184–85
- scalar context, 118–19, 159, 183, 192
- Scalar::Util module, 340
- scaling operation, 271–72, 320–23, 329–30, 338–39, 503–5, 508, 511, 519–20, 523–24, 528–29
- scanning, *see* lexers
- schema, 140, 142, 150
- Schönfinkel, Moses, 327
- Schwartz, Randal, xv, xviii
- Schwern, Michael, 145
- scope, 20, 67–68, 71–76, 92, 98, 264, 454
 - vs. duration, 72–89
- searching databases backwards, 148–53
- secret weapon, 185
- seed (of pseudo-random number generator), 153
- seek() function, 142, 148
- self-referent stream definition, 263–69, 280, 323
- semipredicate function, 107
- semipredicate problem, 107, 124, 163, 170
- sentences (in grammars), 377
- sentential form, 377
- serialization, 69
- set difference, 354–56
- set_prototype() function, 340
- shift() function, 126, 223
- silly examples, 184–85, 346, 456–57
- sin() function, 313–14, 319
- sinh() function, 320
- Sitaker, Kragen, xviii, 453
- Slashdot, 334
- slope, 302, 302, 331
- sort() function, 101, 299, 345
- sorting, 82, 101–6
 - partitions, 224
 - comparator functions, 101–4, 291–92, 350, 353
 - tiebreakers in, 103
 - to obtain canonical form, 219
- soup, fish-dog-and-carrot, 162, 275
- spam, 363
- special cases, 32, 110, 211, 236, 284, 345, 409, 455, 478, 505, 536, 556
- splines, 562
- split() function, 137, 364–65
- sprintf() function, 1, 105–6
- SQL, 163, 168, 471
- sqrt() function, 300, 307
- square, definition in Linogram, 498
- square() function, 83
- srand() function, 155
- srand48() function, 155
- srandom() function, 155
- stack, 50, 55–56, 446–47
 - call stack, 125–26, 229–31
 - explicit simulation of, 38, 242–53
- start symbol, 376–77
- start tag in HTML, 26
- stat() function, 294
- statements as expressions, 424–25
- static variables, 67–68, 124
- Stein, Lincoln, 135
- Storable module, 85, 89
- STORE method, 184–85
- streams, 259, 261–62, 333, 375
 - merging, 270–71, 274–75
- strftime() function, 297
- strict 'vars', 345
- string, 96
- string generator, 440
- string literals, lexing, 449–50
- stringification, 416, 467
- Structure and Interpretation of Computer Programs*, xvii
- StrVal() function in overload module, 467
- stub functions, 71–72, 75–76, 99, 112
- subclass, 25
- subdirectories, 12
- sub-parser, 374
- substr(), 341–42
- subtraction as inverse of addition, 482–83
- subtype definitions, 543
- sum() function, 344–46
- Sussman, Gerald Jay, xvii
- Sybase, 145
- symbolic links, 15, 19, 23–24, 183
 - dangling, 19, 23–24, 183, 325
- symbolic references, 51, 550
- symbol table, xv, 70–71, 99, 112
- symbols in grammars, 376–77
- symbols, terminal vs. nonterminal, 377
- syntactic equivalence of tokens, 436
- syntactic sugar, 122, 123, 172, 259, 374
- syntax, 176, 397
 - bizarre, 180
 - irregularity of Perl's, 453
 - obsession with, 176
- synthetic constraints, 520–21, 522–27
- Synthetic_Constraint_Set class, 522–25
- system administrators, 149
- systems programming, 152

T

- \$^T special variable, 82–83
- \t (tab) character, 450
- tab character, 450
- tac command in Unix, 152
- _tag key in HTML::TreeBuilder, 27
- tags, 26, 30–31, 44, 190
- tai64n, 293, 295
- tail of linked list, 256
- tail-call elimination, 229
- tangent function, 320
- tangent line, 302
- tautology, 508
- tell() function, 148
- Tels, 307, 324, 435
- temperature conversion, 472–75, 483, 485–87
- terminals, 377
- terminal symbol, 377
- terminator pattern, 362–65
- term of arithmetic expression, 394
- terms, 394
- \$textfunc argument, 29, 59–61
- text object in Linogram, 561
- thrashing, 129
- three blind mice, 308
- three-dimensional diagrams in Linogram, 518, 556
- tie() operator, 100, 173, 184–87
- tied hashes, 100, 416
- TIEHANDLE method, 187
- TIESCALAR method, 184
- time() function, 82–83, 112, 113
- Time::HiRes module, 112
- times() function, 113
- to-do list, *see* agenda
- tokens, 55–57, 137, 359–60
 - syntactic equivalence of, 436
- tokenization, *see* lexers
- token-manufacturing function, 371–72, 451
- Torkington, Nat, xviii, 129
- tortoise and hare algorithm, 309

tourists, 76
 Tower of Hanoi, 6–11
 transcendental functions, 313, 409
 transforms, *see* filters and transforms
 treasure, 35–39, 206–11
 trees, 12–14, 26–27, 58, 212, 217,
 377–80, 402–3, 440, 535
 iterator for searching, 225–29
 printing, 233–34, 242–43
 recursion as search of, 215–17
 traversal in various orders, 126
 tricks, 86, 90, 107–8, 120, 137, 160,
 179, 211, 312, 372, 416
 trigonometry, 322, 489–90, 496
 trivial equation (0=0), 502
 trivial stream, 259
 Tuma, Jan J., 324
 tuples in Linogram, 491, 518–20,
 525–30, 555–56, 558
 Type class, 530–39
 Type::Scalar class, 531–32
 %TYPES hash in Linogram, 542–43
 typesetting, 204
 type specimen catalog, 204–5

U

ugly code, 373
 unary minus, 555
 undef, 94, 160, 163–64, 167
 undefined variables in calculator
 program, 426
 union scale, 207
 UNIVERSAL::isa() function, 88, 181,
 258
 Unix, 15, 117, 150, 294
 epoch time format, 295
 untagged text, 27

until loop, 233
 upto(), 121–23, 259–60
 URI::URL module, 199
 URL, 188–93
 bad, 191
 fragment part, 199
 mailto, 200
 referring, 149, 192–93
 user-agent, 198
 user parameter to callback function, 49,
 51, 60–61, 371, 386, 454
 user interfaces, 16
 user-supplied key generators, applications
 of, 89–90

V

Value class, 515–16
 Value::Constant class, 516–18,
 558
 Value::Feature class, 527–30, 535,
 558–59
 Value::Tuple class, 518–20
 value of parsed input, 376, 384–85
 values() function, 119
 variables, static, 67–68
 VERBOSITY directive, 42
 virtual methods, 25
 vline, definition in Linogram,
 497

W

Wadler, Philip, xvii
 Wall, Larry, xv, xviii, 145, 176
 waste of time, 64, 104, 110, 129–30, 190,
 197, 230, 265–68, 304, 307, 461
 web browsers, 68, 108

web robots, *see* web spiders
 web server log, 148–50
 web spiders, 187–201
 overview, 187–90
 DFS unsuitable for, 214
 pursuing only interesting links, 190–92
 referring URLs, 192–97
 robots.txt, 197–200
 \$whence argument to seek(), 142
 while (caller()) loop, 420
 while loop, 13, 187, 209
 reading filehandle in, 187
 that executes only once, 222
 whitespace, 27, 441
 wildcard, 136–37, 136–38
 Windows, 117
 WIRED magazine, 228
 “without” operator, 354–56
 wizard, 206–7
 Wolf Book, 234, 293
 Wong, Meng, xviii
 word boundary, 272, 439
 Wrigley, Ave, 187
 WWW::RobotRules module, 198–99
 WWW::SimpleRobot module, 187–88,
 200–1
 WYSIWYG drawing system, 489

Y

yes command in Unix, 117
 Yukon Territory, 179

Z

0e0 trick, 107–8
 "0 but true", 108