



Index

A

- \a escape sequence, 59
- Abstract accessors, 31, 325–326
- Abstract classes
 - and interfaces, 391
 - overview, 274–275
- Abstract events, 333
- Abstract indexers, 334
- Abstract methods, 25, 313–315
- Access and accessibility
 - array elements, 369
 - constructed types, 495
 - containing types, 284–285
 - events, 160
 - indexers, 161, 179–180
 - members, 19, 79–80, 280
 - accessibility domains, 81–84
 - constraints, 85–86
 - declared accessibility, 80–81
 - generic, 482–483, 524–526, 529–531
 - interface, 378–380
 - pointer, 439–440
 - in primary expressions, 173–176
 - protected, 84–85, 482–483
 - nested types, 282–285
 - pointer elements, 440–441
 - primary expression elements, 178–180
 - properties, 159
- Accessors
 - abstract, 31, 325–326
 - attribute, 417–418
 - event, 331–333
 - property, 31, 319–324, 587
 - declarations, 587–588
 - overriding and interface implementation, 589–590
 - types of, 324–325
 - usage, 588–589
- Acquire semantics, 294
- Acquisition in using statement, 261
- add accessors
 - attributes, 418
 - events, 33, 331
- Add method, 30
- AddEventHandler method, 332
- Addition operator
 - described, 13
 - uses, 202–204
- Address-of operator, 441–442
- Addresses
 - fixed variables, 446–450
 - pointers for, 434–435, 441–442
- after state for enumerator objects, 553–555
- Alert escape sequence, 59

Aliases

- for constructed types, 496
- extern, 596–600
- for namespaces and types, 266–268, 592–594
 - qualified members, 594–595
 - uniqueness of, 595–596

Alignment enumeration, 40

Alloc method, 452–453

Allocation, stack, 450–451

AllowMultiple parameter, 412

Ambiguities, grammar, 504–505

Ampersands (&)

- for addresses, 436
- in assignment operators, 221
- definite assignment rules, 130
- for logical operators, 216, 218–220
- for pointers, 441–442
- in preprocessing expressions, 64

AND operators, 13

Anonymous methods, 463–466

- blocks, 537–538
- conversions, 466, 536–537, 544–545
- definite assignment, 543
- delegate creation expressions, 546
- delegate instance equality, 542
- evaluation, 541–542
- expressions, 535
- implementation example, 546–549
- outer variables, 538–541
- parameters, 464–465
- signatures, 535–536

Applicable function members, 166

Application domains, 73

Applications, 5

- startup, 73–74
- termination, 74

Apply method, 41, 466

Arguments, 21. *See also* Parameters

- command-line, 73
- for function members, 162–165

Arithmetic operators, 198

- addition, 202–204
- division, 199–201
- multiplication, 198–199
- pointer, 443–444
- remainder, 201–202
- shift, 208
- subtraction, 205–207

ArithmeticException class, 200, 409

Arrays and array types, 7–8, 11, 112, 367–368

- access to, 178–179, 369
- for constructed types, 497–498
- covariance, 369–370
- creating, 369
- elements of, 35, 116–117, 369
- with foreach, 247
- initializers, 37, 370–372
- members, 79, 369
- new operator for, 36–37, 184–186
- overview, 35–37
- parameter, 22, 304–307, 484
- pointers with, 448
- syntactic grammar, 667

ArrayTypeMismatchException class, 223, 370, 409

as operator, 216

- generic, 520
- nullable types, 583

Assemblies, 5–6

Assignment

- anonymous methods, 543
- in classes *vs.* structs, 359
- definite. *See* Definite assignment
- fixed size buffers, 607
- iterators, 558
- nullable types, 583

Assignment operators, 13, 221–222

- compound, 224–225
- event, 226
- shift, 533–534
- simple, 222–224

Associativity of operators, 149–151

- Asterisks (*)
 - in assignment operators, 221
 - for comments, 49–50, 611–612
 - for multiplication, 198–199
 - for pointers, 433–435, 439
- At sign characters (@) for identifiers, 52–54
- Atomicity of variable references, 133
- Attribute class, 42, 411
- Attributes, 411
 - classes, 411–414, 426–427
 - compilation of, 421
 - compilation units, 263
 - conditional, 602–603
 - for constructed types, 497
 - instances, 420–421
 - for interoperation, 428
 - overview, 42–43
 - parameters for, 413–414
 - partial types, 568
 - reserved, 422
 - AttributeUsage, 422–423
 - Conditional, 423–427, 602–603
 - Obsolete, 427–428
 - sections for, 415
 - specifications, 414–420
 - syntactic grammar, 669–671
- AttributeUsage attribute, 411–413, 422–423
- Automatic memory management, 95–98
- B**
- \b escape sequence, 59
- Backslash characters (\)
 - for characters, 58–59
 - escape sequence, 59
 - for strings, 60
- Base access, 181
- Base classes, 20
 - of constructed types, 494
 - effective, 510
 - partial types, 569
 - specifications for, 275–277
- Base interfaces
 - inheritance from, 374–375
 - partial types, 569–570
- Base types, 157
- before state for enumerator objects, 553, 555
- Better conversions, 167
- Better function members
 - generic, 526–527
 - parameters in, 167
- Binary operators, 149, 341
 - in ID string format, 628
 - lifted, 581
 - numeric promotions, 155–156
 - overload resolution, 153
 - overloadable, 151–152
 - user-defined, 582
- Bind method, 38
- Binding
 - name, 571
 - type parameter, 479
- BitArray class, 336–337
- Bitwise complement operator, 195
- Blocks
 - anonymous methods, 537–538
 - in declarations, 76–77
 - definite assignment rules, 122
 - exiting, 249
 - in grammar notation, 46
 - invariant meaning in, 172
 - iterator, 551–552
 - in methods, 316
 - reachability of, 231
 - in statements, 232–233
 - for unsafe code, 430
- Bodies
 - classes, 277
 - interfaces, 375
 - methods, 23–24, 316
 - struct, 356
- bool type, 8
 - nullable, 584
 - overview, 109

- Boolean values
 - expressions, 228
 - literals, 55
 - operators
 - conditional logical, 219
 - equality, 212
 - logical, 218
 - in struct example, 365–366
 - BottomToTop property, 468–469
 - Box class, 314
 - Boxed instances, invocations on, 169
 - Boxing, 10, 112
 - in classes *vs.* structs, 360
 - in constraints, 513–515
 - conversions, 112–114
 - implicit, 137
 - for nullable types, 576–577
 - break statement
 - definite assignment rules, 125
 - example, 16
 - for for statements, 245–246
 - overview, 250
 - for switch, 240–241
 - for while, 243
 - yield break, 467, 551, 554–558
 - Buffers, fixed size, 603–607
 - Bugs. *See* Unsafe code
 - Button class, 320, 328–330
 - byte type, 9
- C**
- C# 2.0, 457
 - anonymous methods. *See* Anonymous methods
 - conditional attribute classes, 602–603
 - default value expressions, 601–602
 - extern aliases, 596–600
 - fixed size buffers, 603–607
 - generics. *See* Generics
 - iterators, 467–471, 551–566
 - namespace alias qualifiers, 592–596
 - nullable types. *See* Nullable types
 - partial types, 471–472, 567–571
 - pragma directives, 600–601
 - property accessor accessibility, 587–590
 - static classes, 590–592
 - <c> tag, 614
 - Cache class, 260
 - Callable entities, 399
 - Calling generic methods, 501–502
 - Candidate user-defined operators, 154
 - Captured variables, 466, 538–539
 - Carets (^)
 - in assignment operators, 221
 - for logical operators, 216
 - Carriage-return characters
 - escape sequence, 59
 - as line terminators, 48–49
 - Case labels, 239–242
 - Cast expressions, 197
 - catch blocks, 255–258
 - for exceptions, 408
 - generic, 520
 - char type, 106
 - Character literals, 58–59
 - Characters, 8
 - checked statement
 - definite assignment rules, 122
 - example, 17
 - overview, 258–259
 - in primary expressions, 190–193
 - Chooser class, 502
 - Classes
 - accessibility, 19
 - attribute, 411–414, 426–427
 - base, 20
 - constructed types, 494
 - effective, 510
 - partial types, 569
 - specifications for, 275–277
 - bodies, 277
 - conditional attribute, 602–603
 - constants for, 287–289
 - constructors for, 30–31
 - instance, 343–349
 - static, 349–352
 - declarations, 273
 - base specifications, 275–277

- bodies, 277
- generic. *See* Generics
- modifiers, 273–275
- defined, 273
- destructors for, 34, 352–354
- events in, 32–33
 - accessors, 331–333
 - declaration, 327–329
 - field-like, 329–331
 - instance and static, 332
- fields in, 20–21
 - declarations, 290–291
 - initializing, 295
 - read-only, 292–293
 - static and instance, 291
 - variable initializers, 295–298
 - volatile, 293–295
- function members in, 29–34
- generic. *See* Generics
- indexers in, 32, 333–338
- instance variables in, 116
- interface implementation by, 38
- members in, 18–19, 79, 277–279
 - access modifiers for, 280
 - constituent types for, 280
 - inheritance of, 279
 - nested types for, 281–285
 - new modifier for, 280
 - reserved names for, 286–287
 - static and instance, 280–281
- methods in, 21–28
 - abstract, 313–315
 - bodies, 316
 - declaration, 299–300
 - external, 315–316
 - parameters, 301–307
 - sealed, 313
 - static and instance, 308
 - virtual, 308–311
- modifiers, 273–275
- operators in, 33–34
 - binary, 341
 - conversion, 341–343
 - declaration, 338–339
 - unary, 340–341
- overview, 18
- in program structure, 5
- properties in, 31
 - accessors for, 319–326
 - declarations, 317–318
 - static and instance, 318
- vs.* structs, 357–362
- syntactic grammar, 657–665
- types, 7–9, 11, 110–111
- Classifications, expression, 147–149
- Click events, 329–330
- Closed types, 477, 493
- Coalescing operator, 475, 584–585
- `<code>` tag, 614
- Collections for foreach, 246
- Colons (:)

 - for grammar productions, 46
 - for interface identifiers, 374
 - for namespace alias qualifiers, 593–596
 - for ternary operators, 132, 220

- Color class, 21, 292
- Color enumeration, 39, 393–396
- Color struct, 175–176
- COM, interoperation with, 428
- Combining delegates, 204, 401
- Command-line arguments, 73
- Commas (,)

 - for arrays, 36, 371
 - for attributes, 415
 - in ID string format, 624
 - for interface identifiers, 374

- Comments, 611
 - documentation file processing, 623–629
 - example, 629–634
 - lexical grammar, 49–50, 636
 - overview, 611–613
 - tags, 613–622
 - XML for, 611–612, 632–634
- Commit method, 67–68

- CompareExchange method, 562
- Comparison operators, 33, 209
 - booleans, 212
 - decimal numbers, 211
 - delegates, 214–215
 - enumerations, 212
 - floating point numbers, 210–211
 - integers, 210
 - lifted form, 474
 - pointers, 444
 - reference types, 212–214
 - strings, 214
- Compatibility of delegates and methods, 400, 536
- Compilation
 - attributes, 421
 - conditional, 602–603
 - just-in-time, 6
- Compilation directives, 66–69
- Compilation symbols, 63
- Compilation unit productions, 47
- Compilation units, 45, 263–264
- Compile-time type of instances, 25, 308
- Complement operator, 195
- Complete inferences, 504
- Component-oriented programming, 3
- Compound assignment
 - nullable types, 583
 - overview, 224–225
- Concatenation, string, 204
- Conditional attribute, 423–427, 602–603
- Conditional classes, 426–427
- Conditional compilation directives, 66–69
- Conditional compilation symbols, 63
- Conditional logical operators, 13, 218–220
- Conditional methods, 423–426
- Conditional operator, 13, 220–221
- Consistent inferences, 504
- Consistent methods, 544
- Console class, 324, 590
- Constant class, 25–27
- Constants, 29
 - declarations, 236, 287–289
 - enums for. *See* Enumerations and enum types
 - expressions, 137, 226–228
 - static fields for, 292–293
 - versioning of, 293
- Constituent types, 280
- Constraints
 - accessibility, 85–86
 - generic, 460–462, 506–511
 - member lookup on type parameters, 513
 - satisfying, 511–512
 - type parameters, 513–517
 - partial types, 568–569
 - value type, 507, 512, 574
- Constructed types, 459, 477
 - accessibility, 495
 - alias directives, 496
 - arrays and IList interface for, 497–498
 - attributes, 497
 - base classes and interfaces, 494
 - conversions, 495–496
 - generics, 491–492
 - members, 494–495
 - open and closed, 477, 493
 - type parameters, 493
- Constructors, 29
 - for classes, 30–31
 - for classes *vs.* structs, 361–362
 - default, 103, 348
 - in generic classes, 482
 - in ID string format, 626
 - instance. *See* Instance constructors
 - invocation, 161
 - static, 30, 349–352, 482
- Contexts
 - for attributes, 417–418
 - unsafe, 429–432
- continue statement
 - definite assignment rules, 125
 - for do, 244
 - example, 16
 - for for statements, 245

- overview, 251
 - for while, 243
- Contracts, interfaces as, 373
- Control class, 331–332
- Control-Z character, 48
- Conversions, 135
 - anonymous methods, 466, 536–537, 544–545
 - as operator for, 216
 - boxing, 112–114, 137
 - constant expression, 137
 - constructed types, 495–496
 - enumerations, 136, 140–141
 - explicit, 138–142
 - expressions, 197
 - function members, 167
 - identity, 136
 - implicit, 135–137
 - standard, 142
 - user-defined, 144–145, 578–579
 - lifted, 473–474, 578
 - nullable, 473–474, 574–575
 - boxing and unboxing, 576–577
 - literals, 575
 - user-defined, 577–580
 - numeric, 136, 138–140
 - operators, 341–343, 629
 - for pointers, 437–438
 - reference, 136–137, 141
 - standard, 142
 - type parameter, 515–517
 - unboxing, 114, 142
 - user-defined. *See* User-defined conversions
- Convertible struct, 486–487
- Copy method, 452–453
- Counter class, 323
- Counter struct, 513
- CountPrimes class, 336–337
- Covariance, array, 369–370
- cref attribute, 612
- Critical execution points, 99
- .cs extension, 4

- Curly braces ({})
 - for anonymous methods, 464
 - for arrays, 37, 371
 - in grammar notation, 46
- Current property, 555
- Customer class, 471–472

D

- Database structure example
 - boolean type, 365–366
 - integer type, 363–364
- DBBoolean struct, 365–366
- DBInt struct, 363–364
- Decimal numbers and type, 8–9, 108–109
 - addition, 203
 - comparison operators, 211
 - division, 201
 - multiplication, 199
 - negation, 195
 - remainder operator, 202
 - subtraction, 206
- Declaration directives, 64–66
- Declaration space, 75
- Declaration statements, 234–236
- Declarations
 - classes, 273
 - base specifications, 275–277
 - bodies, 277
 - generic. *See* Generics
 - modifiers, 273–275
 - constants, 236, 287–289
 - definite assignment rules, 123
 - delegates
 - generic, 490–491
 - overview, 399–402
 - enums, 40, 393–394
 - events, 327–329
 - fields, 290–291
 - fixed size buffers, 603–605
 - indexer, 333–338
 - instance constructors, 343–344

- Declarations, *continued*
 - interfaces
 - generic, 488–490
 - overview, 373–375
 - methods, 299–300
 - namespaces, 76–77, 264–265
 - operators, 338–339
 - overview, 75–77
 - parameters, 301–302
 - partial types, 567
 - pointer, 434
 - properties, 317–318
 - property accessors, 319, 587–588
 - static classes, 590–591
 - static constructors, 349–352
 - structs, 355–356
 - generic, 488
 - members, 356–357
 - types, 9, 271–272
 - variables, 234–236
- Declared accessibility
 - nested types, 282–283
 - overview, 80–81
- Decrement operators
 - pointers, 442–443
 - postfix, 181–183
 - prefix, 195–197
- Defaults
 - constructors, 103, 348
 - switch statement labels, 239
 - values, 103, 119
 - classes *vs.* structs, 359–360
 - expressions, 601–602
 - nullable types, 574
- `#define` directive, 63–66
- Definite assignment, 24, 115, 119–120
 - anonymous methods, 543
 - fixed size buffers, 607
 - initially assigned variables, 120–121
 - initially unassigned variables, 121
 - iterators, 558
 - rules for, 121–132
- Delegate class, 399
- delegate keyword, 464
- Delegates and delegate type, 7–9, 11, 112, 399
 - for anonymous methods, 464
 - combining, 204, 401
 - compatible, 400, 536
 - creation expressions for, 546
 - declarations, 399–402, 490–491
 - equality, 214–215, 542
 - generic, 490–491
 - with generic methods, 505–506
 - instantiation, 402–403
 - invocations, 177–178, 403–405
 - members of, 79
 - methods consistent with, 544
 - new operator for, 187–189
 - overview, 40–41
 - removing, 206
 - syntactic grammar, 669
- Delimited comments, 49–50, 611–612
- Dependence
 - on base classes, 276
 - on constraints, 508
 - in structures, 358
- Depends on relationships, 276, 358, 508
- Destructors
 - for classes, 34, 352–354
 - for classes *vs.* structs, 362
 - exceptions for, 409
 - in ID string format, 626
 - member names reserved for, 287
 - members, 19
- Diagnostic directives, 69
- Dictionary class, 460–461, 569
- Digit struct, 342–343
- Direct base classes, 275–276
- Directives
 - extern alias, 598–600
 - pragma, 600–601
 - preprocessing. *See* Preprocessing directives
 - using. *See* Using directives
- Directly depends on relationships, 276, 358
- Disposal in using statement, 261

- Dispose method
 - for enumerator objects, 555, 565–566
 - for resources, 260
 - Divide method, 22
 - DivideByZeroException class, 200–202, 407, 409
 - Division operator, 199–201
 - DllImport attribute, 315
 - DLLs (Dynamic Link Libraries), 315
 - do statement
 - definite assignment rules, 124
 - example, 15
 - overview, 244
 - Documentation comments, 611
 - documentation files for, 611, 623
 - ID string examples, 624–629
 - ID string format, 623–624
 - example, 629–634
 - overview, 611–613
 - tags for, 613–622
 - XML files for, 611–612, 632–634
 - Documentation generators, 611
 - Documentation viewers, 611
 - Domains
 - accessibility, 81–84
 - application, 73
 - Double quotes ("")
 - for characters, 58
 - for strings, 59
 - double type, 8–9, 107
 - Dynamic Link Libraries (DLLs), 315
 - Dynamic memory allocation, 451–453
- E**
- ECMA-334 standard, 3
 - EditBox class, 38
 - Effective base classes, 510
 - Effective interface sets, 511
 - Elements
 - array, 35, 116–117, 369
 - foreach, 247
 - pointer, 440–441
 - primary expression, 178–180
 - #elif directive, 63–64, 66–67
 - Ellipse class, 314
 - #else directive, 63, 66–69
 - Embedded statements and expressions
 - general rules, 128–129
 - in grammar notation, 46
 - Empty statements, 233
 - EmptyStream class, 592–593
 - Encompassed types, 144
 - Encompassing types, 144
 - End points, 230–231
 - #endif directive, 63, 67–69
 - #endregion directive, 70
 - Entity class, 24–25
 - Entry class, 5–6
 - Entry points
 - applications, 73
 - generic classes, 487
 - Enumerable collections, 467
 - Enumerable interfaces, 467, 552
 - Enumerable objects for iterators, 555–556
 - Enumerations and enum types, 393, 397
 - addition of, 203–204
 - comparison operators for, 212
 - conversions
 - explicit, 140–141
 - implicit, 136
 - declarations, 393–394
 - description, 7, 9
 - logical operators for, 217–218
 - members, 79, 394–397
 - modifiers, 394
 - overview, 39–40
 - subtraction of, 206
 - syntactic grammar, 668–669
 - types for, 109
 - values and operations, 397–398
 - Enumerator interfaces, 467, 551
 - Enumerator objects for iterators, 552–555
 - Enumerators, 467
 - Environment class, 590

- Equal signs (=)
 - in assignment operators, 221–222
 - for comparisons, 209
 - operator ==, 33
 - for pointers, 444
 - in preprocessing expressions, 64
- Equality of delegate instances, 542
- Equality operators
 - boolean values, 212
 - delegates, 214–215
 - generic, 519–520
 - lifted, 581
 - nullable types, 582
 - reference types, 212–214, 519–520
 - strings, 214
- Equals method
 - DBBool, 366
 - DBInt, 364
 - List, 30
 - Point, 630–631
- #error directive, 69
- Error property, 324
- Error strings in ID string format, 623
- Escape sequences
 - characters, 58–59
 - lexical grammar, 637
 - strings, 59
 - Unicode character, 51–52
- Evaluate method, 26–27
- Evaluation
 - anonymous methods, 541–542
 - user-defined conversions, 143–144, 577
- Event handlers, 32, 327, 329
- Events, 5
 - access to, 160
 - accessors, 331–333
 - assignment operator, 226
 - declarations, 327–329
 - example, 30
 - field-like, 329–331
 - generic, 506
 - in ID string format, 623, 628
 - instance and static, 332
 - interface, 377
 - member names reserved for, 287
 - overview, 32–33
- <example> tag, 615
- Exception class, 254–255, 407–408
- Exception propagation, 254
- <exception> tag, 615
- Exceptions
 - causes, 407
 - classes for, 409–410
 - for delegates, 403
 - generic, 520
 - handling, 3, 408–409
 - throwing, 254–255
 - try statement for, 255–258
- Exclamation points (!)
 - for comparisons, 209
 - definite assignment rules, 132
 - for logical negation, 195
 - operator !=, 33
 - for pointers, 444
 - in preprocessing expressions, 64
- Execution
 - instance constructors, 346–348
 - order of, 99
- Exiting blocks, 249
- Expanded form function members, 166
- Explicit base interfaces, 374
- Explicit conversions, 138
 - enumerations, 140–141
 - numeric, 138–140
 - reference, 141
 - standard, 142
 - unboxing, 142
 - user-defined, 145–146, 579–580
- Explicit interface member implementations, 38, 381–384, 490
- explicit keyword, 341–343
- Expression class, 25–27
- Expression statements, 15, 122, 236–237
- Expressions, 147
 - anonymous methods, 535
 - boolean, 228
 - cast, 197
 - classifications, 147–149

- constant, 137, 226–228
 - definite assignment rules, 128–132
 - fixed size buffers in, 605–606
 - function members, 157–161
 - argument lists, 162–165
 - invocation, 168–169
 - overload resolution, 165–167
 - generic, 517–521
 - member lookup, 156–167
 - nullable types
 - bool, 584
 - coalescing operator, 584–585
 - compound assignment, 583
 - lifted operators, 580–581
 - user-defined operators, 581–583
 - operators for, 149
 - arithmetic. *See* Arithmetic operators
 - assignment, 221–226
 - logical, 216–220
 - numeric promotions, 154–156
 - overloading, 151–153
 - precedence and associativity, 149–151
 - relational, 209
 - shift, 207–209
 - unary, 193–197
 - overview, 11–13
 - pointers in, 438–446
 - preprocessing, 64
 - primary. *See* Primary expressions
 - syntactic grammar, 647–651
 - values of, 148–149, 601–602
- Extensible Markup Language (XML), 611–612, 632–634
- Extern aliases, 596–600
- External constructors, 344, 350
- External destructors, 352
- External events, 328
- External indexers, 336
- External methods, 315–316
- External operators, 339
- External properties, 318
- F**
- \f escape sequence, 59
 - Factories, enumerator, 470
 - False value, 55, 584
 - Field-like events, 329–331
 - Fields, 5
 - declarations, 290–291
 - example, 29
 - in generic classes, 481–482
 - in ID string format, 623, 625–626
 - initializing, 295, 361
 - instance, 20, 291, 298
 - overview, 20–21
 - read-only, 21, 292–293
 - static, 291–298
 - variable initializers, 295–298
 - volatile, 293–295
 - Fill method, 370
 - Finalize method, 353–354
 - finally blocks
 - for exceptions, 408
 - with goto, 253
 - with try, 255–258
 - Finder class, 499
 - Fixed size buffers, 603
 - declarations, 603–605
 - definite assignment checking, 607
 - in expressions, 605–606
 - fixed statement for, 606
 - fixed statement, 446–450, 606
 - Fixed variables, 436–437
 - float type, 8–9, 107
 - Floating point numbers
 - addition, 203
 - comparison operators, 210–211
 - division, 200
 - multiplication, 198–199
 - negation, 194
 - remainder operator, 201–202
 - subtraction, 205
 - types, 8–9, 107–108

- for statement
 - definite assignment rules, 124–125
 - example, 16
 - overview, 244–246

- foreach statement
 - definite assignment rules, 127
 - example, 16
 - generic, 521
 - for iterators, 467–470
 - overview, 246–248

- Form feed escape sequence, 59

- Fragmentation, heap, 447

- Free method, 452–453

- FromTo method, 560–561

- Fully qualified names
 - described, 94–95
 - interface members, 380
 - nested types, 282

- Function members
 - argument lists, 162–165
 - in classes, 29–34
 - generic, 526–527
 - invocation, 168–169
 - overload resolution, 165–167
 - overview, 157–161

- Function pointers, 399

- Functional notation, 152

G

- Garbage collection, 3
 - at application termination, 74
 - for destructors, 34
 - in memory management, 95–98, 119
 - and pointers, 433
 - for variables, 436

- GC class, 96

- Generics, 458
 - class declarations, 477–478
 - base specifications, 480
 - entry points, 487
 - instance types, 479
 - members, 480–481

- nested types, 487
- operators, 485–487
- overloading, 483–484
- overriding, 484–485
- parameters, 478–479, 484
- protected member access, 482–483
- static constructors, 482
- static fields, 481–482
- constraints, 460–462, 506–511
 - member lookup on type parameters, 513
 - satisfying, 511–512
 - type parameters, 513–517
- constructed types, 491–492
 - accessibility, 495
 - alias directives, 496
 - arrays and IList interface for, 497–498
 - attributes, 497
 - base classes and interfaces, 494
 - conversions, 495–496
 - members, 494–495
 - open and closed, 493
 - type parameters, 493
- creating and using, 459–460
- delegate declarations, 490–491
- expressions and statements, 517–521
- instantiations, 460
- interface declarations, 488
 - member implementations, 490
 - unique implementations, 488–490
- members
 - access to, 482–483, 524–526, 529–531
 - function, 526–527
 - implementations for, 490
 - invocations, 531–533
- methods, 462–463, 498–499
 - calling, 501–502
 - delegates with, 505–506
 - grammar ambiguities, 504–505
 - signatures, 499
 - type argument inference, 502–504
 - virtual, 500–501

- namespaces and type names, 521–524
- purpose of, 458–459
- right-shift operators, 533–534
- simple names, 527–529
- struct declarations, 488
- get accessors
 - for attributes, 417
 - declarations, 587
 - defined, 31
 - working with, 319–326
- GetEnumerator method
 - for foreach, 246
 - for iterators, 467–468, 470, 555–556, 559–564
- GetEventHandler method, 331–332
- GetHashCode method
 - in DBBool, 366
 - in DBInt, 364
- GetNextSerialNo method, 25
- GetProcessHeap method, 452
- Global declaration space, 75
- global identifier, 594–595
- Global namespace, 78, 597
- goto statement
 - definite assignment rules, 125
 - example, 16
 - for switch, 240–241
 - working with, 251–253
- Governing types of switch statements, 239
- Grammars, 45
 - ambiguities, 504–505
 - lexical. *See* Lexical grammar
 - notation, 45–47
 - syntactic. *See* Syntactic grammar
 - for unsafe code, 671–674
- Greater than signs (>)
 - in assignment operators, 221
 - for comparisons, 209
 - for pointers, 435, 439–440, 444
 - for shift operators, 207–209, 533
- Group conversions, 544–545

H

- Handlers, event, 32, 327, 329
- HasValue property, 472–473, 573–574
- Heap
 - accessing functions of, 451–453
 - fragmentation, 447
- HeapAlloc method, 452
- HeapFree method, 452
- HeapReAlloc method, 453
- HeapSize method, 453
- Hello, World program, 4–5
- HelpAttribute class, 42, 413–414
- HelpStringAttribute class, 419
- Hexadecimal escape sequences
 - for characters, 58
 - for strings, 61
- Hiding
 - inherited members, 75, 91–93, 279
 - in multiple-inheritance interfaces, 380
 - in nesting, 90–91, 283
 - properties, 320
 - in scope, 88–93
- Horizontal tab escape sequence, 59

I

- IBase interface, 379–380, 390–391
- ICloneable interface, 381–383, 385
- IComboBox interface, 37, 374–375
- IComparable interface, 381–383, 461–462
- IControl interface, 38, 374, 380, 384–389
- ICounter interface, 378, 514
- ID string format
 - for documentation files, 623–624
 - examples, 624–629
- IDataBound interface, 38
- Identical simple names and type names, 175–176
- Identifiers
 - interface, 374, 377
 - lexical grammar, 637–638
 - rules for, 52–54

- Identity conversions, 136
- IDisposable interface, 247, 260, 382, 521
- IDouble interface, 379
- IEnumerable interface, 247, 467–469, 552
- IEnumerator interface, 467, 551
- #if directive, 63–64, 66–69
- if statement
 - definite assignment rules, 123
 - example, 15
 - working with, 237–238
- IInteger interface, 378–379
- IL (Intermediate Language) instructions, 6
- IList interface, 378, 497–498
- IListBox interface, 37, 374
- IListCounter interface, 378
- IMethods interface, 390–391
- Implicit conversions, 135–137
 - operator for, 341–343
 - standard, 142
 - user-defined, 144–145, 578–579
- implicit keyword, 341–343
- Importing types, 269–271
- In property, 324
- Inaccessible members, 80
- <include> tag, 612, 615–616
- Increment method, 514
- Increment operators
 - for pointers, 442–443
 - postfix, 181–183
 - prefix, 195–197
- IndexerName Attribute, 428
- Indexers
 - access to, 161, 179–180
 - declarations, 333–338
 - example, 29
 - generic, 506
 - in ID string format, 627
 - interface, 377
 - member names reserved for, 287
 - overview, 32
- IndexOutOfRangeException class, 179, 409
- Indices, array, 35–36
- Indirection, pointer, 435, 439
- Inferencing, type, 463, 502–504
- Inheritance
 - in classes, 20, 78, 279
 - in classes *vs.* structs, 359
 - hiding through, 75, 91–93, 279
 - interface, 387–389
 - parameters, 413
 - properties, 320
- Initializers
 - array, 37, 370–372
 - field, 295, 361
 - in for statements, 245
 - instance constructors, 344–345
 - stack allocation, 450–451
 - variables, 295–298, 345
- Initially assigned variables, 115, 120–121
- Initially unassigned variables, 115, 121
- Inlining process, 323
- InnerException property, 408
- Input production, 47
- InputForm class, 464–465
- Instance constructors, 30
 - declarations, 343–344
 - default, 348
 - execution, 346–348
 - initializers, 344–345
 - invocation, 161
 - optional parameters, 349
 - private, 348–349
- Instance events, 332
- Instance fields
 - class, 291
 - example, 20–21
 - initialization, 298
- Instance members
 - class, 280–281
 - protected access for, 84–85
- Instance methods, 21, 24–25, 308
- Instance properties, 318
- Instance types, 479
- Instance variables, 116, 291
- Instances, 18
 - attribute, 420–421
 - type, 110

- Instantiation
 - delegates, 402–403
 - generics, 460
 - local variables, 539–541
- int type, 8
- Integers
 - addition, 202–203
 - comparison operators for, 210
 - division, 199–200
 - literals, 55–56
 - logical operators for, 217
 - multiplication, 198
 - negation, 194
 - remainder, 201
 - in struct example, 363–364
 - subtraction, 205
- Integral types, 8, 105–107
- interface keyword, 373
- Interface sets, effective, 511
- Interfaces, 5, 373
 - base
 - inheritance from, 374–375
 - partial types, 569–570
 - bodies, 375
 - constructed types, 494
 - declarations, 373–375
 - enumerable, 467, 552
 - enumerator, 467, 551
 - generic, 488–490
 - implementations, 380–381
 - abstract classes, 391
 - base classes, 277
 - explicit member, 381–384
 - inheritance, 387–389
 - mapping, 385–387
 - property accessors, 589–590
 - reimplementation, 389–391
 - members, 79, 375–376
 - access to, 378–380
 - events, 377
 - fully qualified names, 380
 - indexers, 377
 - methods, 376–377
 - properties, 377
 - modifiers, 373–374
 - overview, 37–39
 - struct, 356
 - syntactic grammar, 667–668
 - types, 7–9, 11, 111
- Intermediate Language (IL)
 - instructions, 6
- Internal accessibility, 19, 80
- Interoperation attributes, 428
- IntToString method, 451
- IntVector class, 340
- InvalidCastException class, 114, 141, 409, 459, 461, 576
- InvalidOperationException class, 562, 574–575
- Invariant meaning in blocks, 172
- Invocation
 - delegates, 177–178, 403–405
 - function members, 168–169
 - generic members, 531–533
 - instance constructors, 161
 - methods, 158
 - operators, 161
- Invocation expressions, 129, 176–178
- Invocation lists, 401, 403
- IPrintable interface, 511
- is operator
 - generic, 520
 - nullable types, 582–583
 - overview, 215–216
- isFalse property, 365
- isNull property
 - in DBBool, 365
 - in DBInt, 363
- ISO/IEC 23270 standard, 3
- isTrue property, 365
- Iteration statements, 243
 - do, 244
 - for, 244–246
 - foreach, 246–248
 - while, 243–244
- Iteration variables in foreach, 246

Iterators, 467–471
 blocks, 551–552
 definite assignment of, 558
 enumerable objects for, 555–556
 enumerator objects for, 552–555
 implementation example, 558–566
 yield statements for, 556–558
ITextBox interface, 37, 374, 380, 384, 387

J

Jagged arrays, 36
JIT (Just-In-Time) compiler, 6
Jump statements, 248–250
 break, 250
 continue, 251
 goto, 251–253
 return, 253–254
 throw, 254–255
Just-In-Time (JIT) compiler, 6

K

Keywords
 lexical grammar, 638
 list, 54–55

L

Label class, 321–322
Label declaration space, 76
Labeled statements
 for goto, 251–252
 overview, 233–234
 for switch, 239–242
Left-associative operators, 150
Left shift operator, 208
Length of arrays, 36, 367, 371–372
Less than signs (<)
 in assignment operators, 221
 for comparisons, 209
 for pointers, 444
 for shift operators, 207–209, 533–534
Lexical grammar, 47–48, 635
 comments, 49–50, 636

 identifiers, 637–638
 keywords, 638
 line terminators, 48–49, 635
 literals, 639–641
 operators and punctuators, 641
 preprocessing directives, 641–644
 tokens, 637
 Unicode character escape sequences, 637
 whitespace, 49, 637
Lexical structure, 45
 grammars, 45–47
 lexical. *See* Lexical grammar
 syntactic. *See* Syntactic grammar
 lexical analysis, 47–50
 preprocessing directives, 61–63
 conditional compilation, 63, 66–69
 declaration, 64–66
 diagnostic, 69
 line, 70–71
 preprocessing expressions, 64
 region, 69–70
 programs, 45
 tokens, 51
 identifiers, 52–54
 keywords, 54–55
 literals, 55–61
 operators, 61
 Unicode character escape sequence,
 51–52
Libraries, 5, 315
Lifted conversions, 473–474, 578
Lifted operators, 473, 580–581
#line directive, 70–71
#line default directive, 71
Line directives, 70–71
Line-feed characters, 48
#line hidden directive, 71
Line-separator characters, 48
Line terminators, 48–49, 635
List class, 29–33
<list> tag, 616–617
ListChanged method, 33
Lists, statement, 232–233

- Literals, 55
 - boolean, 55
 - character, 58–59
 - in constant expressions, 226
 - integer, 55–56
 - lexical grammar, 639–641
 - null, 61
 - nullable conversions, 575
 - in primary expressions, 170
 - real, 57
 - simple values, 105
 - string, 59–61
 - Local constant declarations, 15, 236
 - Local variable declaration space, 76
 - Local variables, 118–119
 - declarations, 14, 234–236
 - instantiation, 539–541
 - in methods, 23–24
 - scope, 90
 - lock statement
 - definite assignment rules, 128
 - example, 17
 - generic, 520
 - overview, 259–260
 - Logical operators, 216
 - AND, 13
 - for boolean values, 218
 - conditional, 218–220
 - for enumerations, 217–218
 - for integers, 217
 - negation, 195
 - OR, 13
 - shift, 208
 - XOR, 13
 - LoginDialog class, 328–329
 - long type, 8
 - Lookup, member, 156–167, 513
 - lvalues, 133
- M**
- Main method
 - for startup, 73–74
 - for static constructors, 350–352
 - Mappings
 - interface, 385–387
 - pointers and integers, 438
 - Members, 5, 18–19, 77–78
 - access to, 19, 79–80, 280
 - accessibility domains, 81–84
 - constraints, 85–86
 - declared accessibility, 80–81
 - interface, 378–380
 - pointer, 439–440
 - in primary expressions, 173–176
 - protected, 84–85, 482–483
 - accessibility of, 19
 - array, 79, 369
 - class, 79, 277–279
 - access modifiers for, 280
 - constituent types for, 280
 - inheritance of, 279
 - nested types for, 281–285
 - new modifier for, 280
 - reserved names for, 286–287
 - static and instance, 280–281
 - constructed types, 494–495
 - delegate, 79
 - enumeration, 79, 394–397
 - function. *See* Function members
 - generic, 480–481
 - access to, 482–483, 524–526, 529–531
 - function, 526–527
 - implementations for, 490
 - invocations, 531–533
 - inherited, 75, 78, 91–93, 279
 - interface, 79, 375–376
 - access to, 378–380
 - events, 377
 - explicit implementations of, 38, 381–384
 - fully qualified names, 380
 - indexers, 377
 - methods, 376–377
 - properties, 377
 - lookup, 156–167, 513

Members, *continued*

- namespaces, 78, 271
- nullable types, 573–574
- partial types, 570
- pointer, 439–440
- qualified alias, 594–595
- struct, 78, 356–357

Memory

- automatic management of, 95–98, 119
- dynamic allocation of, 451–453

Memory class, 451–453

Message property, 408

Metadata, 6

Methods, 5, 21

- abstract, 25, 313–315
- anonymous. *See* Anonymous methods
- bodies, 23–24, 316
- conditional, 423–426
- declarations, 299–300
- external, 315–316
- generic, 462–463, 498–499
 - calling, 501–502
 - delegates with, 505–506
 - grammar ambiguities, 504–505
 - signatures, 499
 - type argument inference, 502–504
 - virtual, 500–501
- in ID string format, 623, 626–627
- instance, 21, 24–25, 308
- interface, 376–377
- invocations, 158, 176–177
- in List, 30
- overloading, 28
- overriding, 25, 311–312
- parameters, 21–23
 - arrays, 304–307
 - declarations, 301–302
 - generic list, 498–499
 - output, 303–304
 - reference, 302–303
 - value, 302
- sealed, 313
- static, 21, 24–25, 308

- virtual, 25–28, 308–311

Minus (-) operator, 194–195

Minus signs (-)

- in assignment operators, 221–222
- for decrement operator, 181–183, 195–197
- for pointers, 435, 439–440, 442–444
- for subtraction, 205–207

Modifiers

- class, 273–275
- enums, 394
- interface, 373–374
- partial types, 568
- struct, 356

Most derived method implementation, 309

Most encompassing types, 144

Most specific operators, 143

Move method, 630

Moveable variables

- described, 436–437
- fixed addresses for, 446–450

MoveNext method, 246, 553–554, 560, 564–565

Multi-dimensional arrays, 10, 36, 367, 371

Multi-use attribute classes, 412

multiple inheritance, 37, 380

Multiple statements, 232

Multiplication operator, 12, 198–199

Multiplicative operators, 12

Multiplier class, 41

Multiply method, 41

MultiplyAllBy method, 465–466

Mutual-exclusion locks, 259–260

N

\n escape sequence, 59

Named constants. *See* Enumerations and enum types

Named parameters, 413–414

Names

- binding, 571
- fully qualified, 94–95
- interface members, 380
- nested types, 282

- Names, *continued*
 - hiding, 90–93
 - reserved, 286–287
 - simple
 - generic, 527–529
 - in primary expressions, 171–173
 - and type names, 175–176
- namespace keyword, 264
- Namespaces, 4–5, 93–94, 263
 - aliases for, 266–268, 592–594
 - qualified members, 594–595
 - uniqueness of, 595–596
 - compilation units, 263–264
 - declarations, 76–77, 264–265
 - fully qualified names in, 94–95
 - generic, 521–524
 - in ID string format, 623
 - members, 78, 271
 - syntactic grammar, 656–657
 - type declarations, 271–272
 - using directives in, 265–271
- NaN (Not-a-Number) value
 - causes, 107
 - in floating point comparisons, 211
- Negation
 - logical, 195
 - numeric, 194–195
- Nested array initializers, 371
- Nested blocks, 77
- Nested classes, 274
- Nested members, 81–82
- Nested scopes, 87–90
- Nested types, 272, 281–282
 - accessibility of, 282–283
 - fully qualified names for, 282
 - in generic classes, 487
 - member access contained by, 284–285
 - this access to, 283–284
- Nesting
 - aliases, 267–268
 - with break, 250
 - comments, 50
 - hiding through, 90–91, 283
- New line escape sequence, 59
- new modifier
 - class members, 280
 - classes, 274
 - delegates, 400
 - interfaces, 374
- new operator
 - arrays, 36–37, 184–186
 - constructors, 31
 - delegates, 187–189
 - objects, 183–184
 - structs, 35
- No fall through rule, 240–241
- No side effects convention, 323
- Non-nested types, 281
- Non-virtual methods, 25
- Nonterminal symbols, 45–46
- Normal form function members, 166
- Normalization Form C, 53
- Not-a-Number (NaN) value
 - causes, 107
 - in floating point comparisons, 211
- Notation, grammar, 45–47
- NotSupportedException class, 553
- Null field for events, 32
- Null literal, 61
- Null pointers, 434
- Null propagating conversion form, 473
- Null-termination of strings, 449–450
- Null values
 - for array elements, 37
 - in bool type, 584
 - in classes *vs.* structs, 359–360
 - escape sequence for, 59
 - garbage collector for, 97
- Nullable types, 472–475, 573
 - conversions, 473–474, 574–575
 - boxing and unboxing, 576–577
 - literals, 575
 - user-defined, 577–580

Nullable types, *continued*

- default values, 574
- expressions
 - bool, 584
 - coalescing operator, 475, 584–585
 - compound assignment, 583
 - lifted operators, 580–581
 - user-defined operators, 581–583
- members, 573–574
- value type constraints, 574

NullReferenceException class, 114, 178–179, 246, 403, 409

Numeric conversions

- explicit, 138–140
- implicit, 136

Numeric promotions, 154–156

O

object class, 102, 111

Object variables, 11

Objects

- creation expressions for
 - definite assignment rules, 129
 - generic, 517
 - new operator, 183–184
- as instance types, 110

Obsolete attribute, 427–428

OnChanged method, 30, 32

One-dimensional arrays, 36

Open types, 477, 493

Operands, 11, 149

Operation class, 26

Operator notation, 152

Operators, 11, 30, 33–34, 61, 149

- arithmetic. *See* Arithmetic operators
- assignment. *See* Assignment operators
- binary. *See* Binary operators
- conditional, 220–221
- conversion, 341–343, 629
- declaration, 338–339
- enums, 397–398
- generic, 485–487, 506
- in ID string format, 628–629

invocation, 161

lexical grammar, 641

lifted, 473, 580–581

logical, 216–220

nullable types, 581–583

numeric promotions, 154–156

operator ==, 33

operator !=, 33

overloading, 151–153

precedence and associativity, 149–151

relational. *See* Relational operators

shift, 207–209

type-testing, 215–216

unary. *See* Unary operators

Optional parameters, 349

Optional symbols in grammar notation, 46

OR operators, 13

Order

declaration, 76

execution, 99

Out property, 324

Outer variables, 538

captured, 538–539

defined, 466

instantiation, 539–541

OutOfMemoryException class, 186, 188, 204, 409

Output parameters, 22, 117–118, 303–304

Overflow checking context, 190–193, 258–259

OverflowException class, 109, 191–192, 196, 199, 201, 410

Overload resolution

function members, 165–167

operators, 582

Overloaded operators, 11

purpose, 149

shift, 207

Overloading

in generic classes, 483–484

indexers, 32

methods, 28

operators, 151–153

signatures in, 28, 86–87

Overridden base methods, 311

- Override events, 333
- Override indexers, 334
- Override methods, 311–312
- Overriding
 - event declarations, 333
 - in generic classes, 484–485
 - methods, 25
 - property accessors, 31, 325, 589–590
 - property declarations, 325
- P**
- Padding for pointers, 445
- Paint method, 38, 314
- Pair-wise declarations, 341
- <para> tag, 617–618
- Paragraph-separator characters, 48
- <param> tag, 612, 618
- Parameters
 - arrays, 304–307, 484
 - attributes, 413–414
 - entry points, 73
 - indexers, 32, 335
 - instance constructors, 345, 349
 - for methods, 21–23
 - anonymous, 464–465
 - declaration, 301–302
 - generic list, 498–499
 - types of, 302–307
 - optional, 349
 - output, 117–118, 303–304
 - reference, 117, 302–303, 511
 - type. *See* Type parameters
 - value, 117, 302
- <paramref> tag, 618–619
- params modifier, 22, 304–307
- Parentheses ()
 - in grammar notation, 46
 - in ID string format, 623
- Parenthesized expressions, 173
- partial modifier, 471–472, 567
- Partial types, 471–472
 - attributes, 568
 - base classes, 569
 - base interfaces, 569–570
 - declarations, 567
 - members, 570
 - modifiers, 568
 - name binding, 571
 - type parameters and constraints, 568–569
- Percent signs (%)
 - in assignment operators, 221
 - for remainder operator, 202
- periods (.) for base access, 181
- <permission> tag, 619
- Permitted user-defined conversions, 143
- Plus (+) operator, 193–194
- Plus signs (+)
 - for addition, 202–204
 - in assignment operators, 221–222
 - for increment operator, 181–183, 195–197
 - for pointers, 442–444
- Point class, 34
 - base class, 20
 - declaration, 18
 - source code, 629–632
- Point struct, 35, 223–224, 358–359, 361
- Point3D class, 20
- Pointers, 429
 - arithmetic for, 443–444
 - conversions for, 437–438
 - element access, 440–441
 - in expressions, 438–446
 - for fixed variables, 446–450
 - function, 399
 - indirection with, 435, 439
 - member access, 439–440
 - operators
 - address-of, 441–442
 - comparison, 444
 - increment and decrement, 442–443
 - sizeof, 444–445
 - types, 433–436
 - variables with, 436–437
- Pop method, 5–6, 458

- Positional parameters, 413–414
 - Postfix increment and decrement operators, 181–183
 - pragma directives, 600–601
 - Precedence of operators, 11, 149–151
 - Prefix increment and decrement operators, 195–197
 - Preprocessing directives, 61–63
 - conditional compilation, 63, 66–69
 - declaration, 64–66
 - diagnostic, 69
 - lexical grammar, 641–644
 - line, 70–71
 - pragma, 600–601
 - preprocessing expressions, 64
 - region, 69–70
 - Preprocessing expressions, 64
 - Primary constraints, 507
 - Primary expressions
 - checked and unchecked operators, 190–193
 - element access, 178–180
 - invocation, 176–178
 - literals in, 170
 - member access, 173–176
 - new operator in, 183–189
 - parenthesized, 173
 - postfix increment and decrement operators, 181–183
 - simple names in, 171–173
 - this access in, 180–181
 - typeof operator, 189–190
 - Primary operators, 12
 - PrintColor method, 39
 - Private accessibility, 19, 80
 - Private constructors, 348–349
 - Productions, grammar, 45
 - Program class, 514–515, 563
 - Program structure, 5–7
 - Programs, 5, 45
 - Promotions, numeric, 154–156
 - Propagation, exception, 254
 - Properties, 5
 - access to, 159
 - declarations, 317–318
 - example, 29
 - generic, 506
 - in ID string format, 623, 627
 - indexers, 335
 - interface, 377
 - member names reserved for, 286–287
 - overview, 31
 - static and instance, 318
 - Property accessors, 31, 319–324, 587
 - declarations, 319, 587–588
 - overriding and interface implementation, 589–590
 - overview, 319–324
 - types of, 324–325
 - usage, 588–589
 - Protected accessibility, 19, 80
 - generic class members, 482–483
 - instance members, 84–85
 - Protected internal accessibility, 19, 80
 - Public accessibility, 19, 80
 - Punctuators
 - lexical grammar, 641
 - list of, 61
 - PurchaseTransaction class, 67–68
 - Push method, 5–6, 458
 - PushMultiple method, 462–463
- Q**
- Qualified alias members, 594–595
 - Question marks (?)
 - for bool type, 584
 - for coalescing operator, 584–585
 - for nullable types, 573, 576–578
 - for ternary operators, 132, 220
 - for type modifiers, 472–475
- R**
- \r escape sequence, 59
 - Rank of arrays, 36, 367–368

- Reachability
 - blocks, 231
 - do statements, 244
 - for statements, 246
 - labeled statements, 234
 - overview, 230–231
 - throw statements, 254
 - while statements, 243–244
- Read-only fields, 21, 292–293
- Read-only properties, 31, 320–323
- Read-write properties, 31, 320–322
- readonly modifier, 21, 292
- Reads, volatile, 294
- Real literals, 57
- ReAlloc method, 452
- Recommended tags for comments, 613–622
- Rectangle struct, 224
- ref modifier, 22
- Reference conversions
 - explicit, 141
 - implicit, 136–137
- Reference parameters, 22, 117, 302–303
- Reference types, 7–8, 110
 - array, 36, 111
 - class, 110–111
 - constraints, 507, 512
 - delegate, 111
 - equality operators, 212–214, 519–520
 - interface, 111
 - object, 111
 - string, 111
 - type parameters, 511
- References, variable, 133
- Referencing static classes, 591–592
- Referent types, pointer, 433
- Region directives, 69–70
- Regular string literals, 59–60
- Reimplementation, interface, 389–391
- Relational operators, 13, 209
 - booleans, 212
 - decimal numbers, 211
 - delegates, 214–215
 - enumerations, 212
 - integers, 210
 - lifted, 581
 - reference types, 212–214
 - strings, 214
- Release semantics, 294
- Remainder operator, 201–202
- <remarks> tag, 619
- remove accessors
 - for attributes, 417–418
 - for events, 33, 331
- RemoveEventHandler method, 332
- Removing delegates, 206
- Reserved attributes, 422
 - AttributeUsage, 422–423
 - Conditional, 423–427
 - Obsolete, 427–428
- Reserved names for class members, 286–287
- Reset method, 565
- Resolution
 - function members, 165–167
 - operator overload, 28, 153, 582
- Resources, using statement for, 260–262
- return statement
 - blocks, 538
 - definite assignment rules, 125
 - example, 16
 - methods, 24
 - overview, 253–254
- Return type
 - delegates, 465
 - entry points, 74
 - methods, 21, 300
- <returns> tag, 620
- Right-associative operators, 151
- Right shift operator
 - generics, 533–534
 - overview, 208
- Rules for definite assignment, 121–132
- running state for enumerator objects, 553, 555

- Runtime processes
 - array creation, 185
 - attribute instance retrieval, 421
 - delegate creation, 188
 - function member invocations, 163, 168
 - increment and decrement operators, 182
 - object creation, 184
 - type parameter binding, 479
 - unboxing conversions, 114
- Runtime types, 25, 308

S

- sbyte type, 8
- Scopes
 - aliases, 267–268
 - attributes, 417
 - local variables, 235
 - for name hiding, 90–93
 - overview, 87–90
 - type parameters, 478
- Sealed accessors, 325
- Sealed classes, 275, 277
- Sealed events, 333
- Sealed indexers, 334
- Sealed methods, 313
- sealed modifier, 275, 313
- Secondary constraints, 507
- Sections for attributes, 415
- <see> tag, 620
- <seealso> tag, 621
- Selection statements, 237
 - if, 237–238
 - switch, 238–243
- Semicolons (;)
 - for interface identifiers, 377
 - for method bodies, 316
 - in namespace declarations, 264
- set accessors, 325
 - declarations, 587
 - defined, 31
 - working with, 319–321
- SetNextSerialNo method, 25
- Shape class, 314
- Shift operators
 - described, 13
 - generics, 533–534
 - overview, 207–209
- Short-circuiting logical operators, 218
- short type, 8
- ShowHelp method, 43
- Side effects
 - with accessors, 323
 - and execution order, 99
- Signatures
 - indexers, 335
 - methods, 21
 - anonymous, 535–536
 - generic, 499
 - operators
 - binary, 341
 - conversion, 342
 - unary, 340
 - in overloading, 28, 86–87
- Signed integrals, 8
- Simple assignment
 - definite assignment rules, 129
 - overview, 222–224
- Simple expression assignment rules, 128
- Simple names
 - generic, 527–529
 - in primary expressions, 171–173
 - and type names, 175–176
- Simple types, 7, 101–102, 104–105
- Single-dimensional arrays
 - defined, 367
 - example, 36
 - initializers, 371
- Single-line comments, 49–50, 611–612
- Single quotes (') for characters, 58
- Single-use attribute classes, 412
- SizeOf method, 452
- sizeof operator, 444–445
- Slashes (/)
 - in assignment operators, 221
 - for comments, 49–50, 611–612
 - for division, 199–201

- Source files
 - compilation, 7
 - described, 45
 - Point class, 629–632
- Source types in conversions, 143
- SplitPath method, 304
- Square brackets ([])
 - for arrays, 10, 36
 - for attributes, 415
 - for indexers, 32
 - for pointers, 440–441
- Square method, 41
- Squares class, 23
- Stack allocation, 450–451
- Stack class, 5–6, 458–460, 467–468, 559–560
- stackalloc method, 450–451
- StackOverflowException class, 410, 450
- Standard conversions, 142
- Startup, application, 73–74
- Statement lists, 232–233
- Statements
 - blocks in, 232–233
 - checked and unchecked, 258–259
 - declaration, 234–236
 - definite assignment rules, 122
 - empty, 233
 - end points and reachability, 230–231
 - expression, 15, 122, 236–237
 - generic, 517–521
 - in grammar notation, 46
 - iteration, 243
 - do, 244
 - for, 244–246
 - foreach, 246–248
 - while, 243–244
 - jump, 248–250
 - break, 250
 - continue, 251
 - goto, 251–253
 - return, 253–254
 - throw, 254–255
 - labeled, 233–234
 - lock, 259–260
 - overview, 14–17
 - selection, 237
 - if, 237–238
 - switch, 238–243
 - syntactic grammar, 651–656
 - try, 255–258
 - using, 260–262
- States, definite assignment, 121
- Static classes, 590
 - declarations, 590–591
 - referencing, 591–592
- Static constructors, 30, 349–352
 - in classes *vs.* structs, 362
 - in generic classes, 482
- Static events, 332
- Static fields, 20, 291
 - for constants, 292–293
 - in generic classes, 481–482
 - initialization, 295–298
- Static members, 280–281
- Static methods, 21, 24–25, 308
- Static properties, 318
- Static variables, 116, 291
- Status codes, termination, 74
- Stream class, 593
- string type, 8, 111
- Strings
 - concatenation, 204
 - equality operators, 214
 - literals, 59–61
 - null-termination, 449–450
 - switch governing type, 242
- Structs, 355
 - assignment, 359
 - boxing and unboxing, 360
 - vs.* classes, 357–362
 - constructors, 361–362
 - declarations, 355–356
 - default values, 359–360
 - destructors, 362

Structs, *continued*

- examples
 - database boolean type, 365–366
 - database integer type, 363–364
 - field initializers in, 361
 - generic, 488
 - inheritance, 359
 - instance variables, 116
 - interface implementation by, 38
 - members, 78, 356–357
 - overview, 34–35
 - syntactic grammar, 665–666
 - this access in, 360–361
 - types, 7, 9, 104
 - value semantics, 357–358
- Subtraction operator, 205–207
- <summary> tag, 612, 621
- SuppressFinalize method, 74
- suspended state, 553, 555
- Swap method, 22
- switch statement
 - definite assignment rules, 123
 - example, 15
 - overview, 238–243
 - reachability, 231
- Syntactic grammar, 47
- arrays, 667
 - attributes, 669–671
 - basic concepts, 644–645
 - classes, 657–665
 - delegates, 669
 - enums, 668–669
 - expressions, 647–651
 - interfaces, 667–668
 - namespaces, 656–657
 - statements, 651–656
 - structs, 665–666
 - types, 645–646
 - variables, 646
- system-level exceptions, 407
- System namespace, 104

T

- \t escape sequence, 59
- Tab escape sequence, 59
- Tags for comments, 613–622
- Target types in conversions, 143
- Targets
 - goto, 252
 - jump, 249
- Terminal symbols, 45–46
- Termination, application, 74
- Terminators, line, 48–49, 635
- Ternary operators, 149, 220–221
- TextReader class, 262
- TextWriter class, 262
- this access
 - classes *vs.* structs, 360–361
 - indexers, 32
 - instance constructors, 349
 - iterator blocks, 552
 - nested types, 283–284
 - outer variables, 538
 - overview, 180–181
 - properties, 318
 - static methods, 24
- Three-dimensional arrays, 36
- Throw points, 254
- throw statement
 - definite assignment rules, 125
 - example, 17
 - for exceptions, 407
 - generic, 520
 - overview, 254–255
- Tildes (~)
 - for bitwise complement, 195
 - for conversion, 629
- Tokens, 51
 - identifiers, 52–54
 - keywords, 54–55
 - lexical grammar, 637
 - literals, 55–61
 - operators, 61
 - unicode character escape sequence, 51–52

top method, 6
 TopToBottom property, 468–469
 ToString method, 204

- DBBool, 366
- DBInt, 364
- Point, 631

 Translate method, 630
 Tree class, 562–563
 Trig class, 349
 True value, 55, 584
 try statement

- definite assignment rules, 125–127
- example, 17
- for exceptions, 408
- with goto, 253
- overview, 255–258

 Two-dimensional arrays, 36
 Type arguments, 493

- inference, 502–504

 Type casts, 40
 Type class, 518
 Type names, 93–94

- fully qualified, 94–95
- generic, 521–524
- identical, 175–176

 Type parameters

- boxing, 513–515
- constructed types, 493
- conversions, 515–517
- generic classes, 459, 478–479, 484
- generic methods, 498–499
- member lookup on, 513
- partial types, 568–569

 Type-safe design, 3
 Type testing operators

- as, 216
- described, 13
- is, 215–216

 TypeInitializationException class, 408, 410
 typeof operator

- generic expressions, 517–519

- pointers with, 433
- primary expressions, 189–190

<typeparam> tag, 622
 <typeparamref> tag, 622
 Types, 101

- aliases for, 266–268, 592–594
 - qualified members, 594–595
 - uniqueness of, 595–596
- attribute parameter, 414
- boxing and unboxing, 112–114
- declarations, 9, 271–272
- generic. *See* Generics
- in ID string format, 623–625
- importing, 269–271
- inferencing, 463, 502–504
- nested, 272, 281–285
- nullable. *See* Nullable types
- overview, 7–10
- partial, 471–472
- reference. *See* Reference types
- syntactic grammar, 645–646
- value. *See* Value types

U

uint type, 9
 ulong type, 9
 Unary operators, 193

- cast expressions, 197
- described, 12, 149
- generic classes, 485
- in ID string format, 628
- lifted, 580
- minus, 194–195
- numeric promotions, 154–155
- overload resolution, 153
- overloadable, 151–152
- overview, 340–341
- plus, 193–194
- prefix increment and decrement, 195–197
- user-defined, 581

 Unassigned variables, 121

- Unboxing conversions
 - described, 142
 - for nullable types, 576–577
 - overview, 114
 - Unboxing operations
 - in classes *vs.* structs, 360
 - example, 10
 - unchecked statement
 - definite assignment rules, 122
 - example, 17
 - overview, 258–259
 - in primary expressions, 190–193
 - #undef directive, 63–66
 - Undefined conditional compilation symbols, 63
 - Underlying types
 - enums, 39–40, 393
 - nullable, 573
 - Underscore characters (`_`) for identifiers, 52–54
 - Unicode characters
 - escape sequence, 51–52
 - lexical grammar, 47, 637
 - for strings, 8
 - Unicode Normalization Form C, 53
 - Unified type system, 3
 - Uniqueness
 - aliases, 595–596
 - generic interface implementation, 488–490
 - Unmanaged types, 433
 - Unreachable statements, 230
 - Unsafe code, 429
 - contexts in, 429–432
 - dynamic memory allocation, 451–453
 - fixed statement, 446–450
 - grammar extensions for, 671–674
 - pointers
 - conversions for, 437–438
 - in expressions, 438–446
 - types, 433–436
 - stack allocation, 450–451
 - unsafe modifier, 429–432
 - Unsigned integral, 9
 - Unwrapping nullable type conversions, 575
 - User-defined conversions, 143
 - evaluation, 143–144, 577
 - explicit, 142, 145–146, 579–580
 - implicit, 137, 144–145, 578–579
 - lifted form, 473, 578
 - overview, 341–343
 - permitted, 143, 577
 - User-defined operators
 - candidate, 154
 - conditional logical, 219–220
 - nullable types, 581–582
 - ushort type, 9
 - Using directives
 - for aliases, 266–268
 - definite assignment rules, 128
 - example, 17
 - generic, 521
 - for importing types, 269–271
 - for name binding, 571
 - overview, 260–262, 265–266
 - purpose, 4
- V**
- `\v` escape sequence, 59
 - Value method, 363
 - Value parameters, 22, 117, 302
 - Value property, 472–473, 573–574
 - <value> tag, 621–622
 - Value types, 101–102
 - bool, 109
 - constraints, 507, 512, 574
 - contents, 11
 - decimal, 108–109
 - default constructors, 103
 - described, 7
 - enumeration, 109
 - floating point, 107–108
 - integral, 105–107
 - simple, 104–105
 - struct, 104

Values

- array types, 368
 - classes *vs.* structs, 357–358
 - default, 103, 119
 - classes *vs.* structs, 359–360
 - expressions, 601–602
 - nullable types, 574
 - enums, 397–398
 - of expressions, 148–149, 601–602
 - fields, 291
 - local constants, 236
 - variables, 115, 235
- ValueType class, 102–103, 359
- VariableReference class, 25–27
- Variables, 115
- array elements, 116–117
 - declarations, 234–236
 - default values, 119
 - definite assignment. *See* Definite assignment
 - fixed addresses for, 446–450
 - fixed and moveable, 436–437
 - initializers, 295–298, 345
 - instance, 116, 291
 - instantiation, 539–541
 - local, 118–119
 - in methods, 23–24
 - outer, 466, 538–541
 - output parameters, 117–118
 - overview, 10–11
 - reference parameters, 117
 - references, 133
 - scope, 90, 235
 - static, 116, 291
 - syntactic grammar, 646
 - value parameters, 117
- Verbatim identifiers, 53
- Verbatim string literals, 59–60
- Versioning
- of constants, 293
 - described, 3

Vertical bars (|)

- in assignment operators, 221
 - definite assignment rules, 131
 - for logical operators, 216, 218–220
 - in preprocessing expressions, 64
- Vertical tab escape sequence, 59
- Viewers, documentation, 611
- Virtual accessors, 31, 325
- Virtual events, 333
- Virtual indexers, 334
- Virtual methods
- generic, 500–501
 - overview, 25–28
 - working with, 308–311
- Visibility in scope, 90
- void type and values
- entry point method, 74
 - events, 331
 - pointers, 434
 - return, 21, 24
 - typeof, 518
- Volatile fields, 293–295

W

- WaitForPendingFinalizers method, 98
- #warning directive, 69, 600–601
- where keyword, 461, 511
- while statement
- definite assignment rules, 124
 - example, 15
 - overview, 243–244
- Whitespace
- in comments, 612
 - defined, 49
 - in ID string format, 623
 - lexical grammar, 637
- Win32 component interoperability, 428
- Wrapping nullable type conversions, 574
- Write method, 23
- Write-only properties, 31, 320–322
- WriteLine method, 5, 23
- Writes, volatile, 294

X

XAttribute class, 419
XML (Extensible Markup Language),
611–612, 632–634
XOR operators, 13

Y

yield statements
for iterators, 556–558
yield break, 467, 551, 554–558
yield return, 467, 551, 554–558