

# Index

## SYMBOLS AND NUMERICS

- \* (asterisk), in regular expressions, 470
- [] (brackets), in regular expressions, 465, 470, 471
- ^ (caret), in regular expressions, 475
- { } (curly braces), in regular expressions, 465
- .
- (dot) class, in regular expressions, 471
- () (parentheses), in regular expressions, 477–479
- | (pipe character), in regular expressions, 474
- + (plus symbol), in regular expressions, 470
- ? (question mark), in regular expressions, 474, 475
- \0 escape, in regular expressions, 468
- 2PC (two-phase commit), 548
- 64-bit architecture
  - JIT (just-in-time) compilation supporting, 131
  - version 2.0 improvements for, 7

## A

- \a escape, in regular expressions, 468
- \A meta-character, in regular expressions, 475
- Aborted state, Thread object, 358
- AbortRequested state, Thread object, 358
- abstract classes, 51–52, 56–57
- abstract methods, 51
- Accept method, sockets, 283, 288–289
- access control, 273, 348–351. *See also* CAS (code access security)
- access violation (AV), 12
- accessibility of assembly, 25
- accessibility of types, 25–26, 45
- AccessViolationException exception, 210
- ACEs (Access Control Entries), 349

- ACID properties of transactions, 535
- ACLs (Access Control Lists), 348–351, 374
- action functions, 252–253
- Action<T> delegate, 238, 252–253
- activation frame, 126
- activation, reflection, 496
- Active Server Pages (ASP), 4
- add instruction, IL, 84–85, 91, 550
- AddMemoryPressure method, GC class, 203–204
- ADO.NET, 5, 540–541
- allocation of memory, 115–120
- ALS (AppDomain-Local Storage), 397
- alternations, in regular expressions, 474–475
- always-sorted dictionaries, 241
- ambient transactions, 536
- American Standard Code for Information Interchange (ASCII) encoding, 320
- and instruction, IL, 91, 550
- anonymous delegates, 63–64
- anonymous methods, 527–529
- AOP (Aspect Oriented Programming), 440
- apartments, COM, 390–391, 402
- APIs (application program interfaces)
  - builder APIs, 529–532
  - info APIs
    - assembly information, 498–500
    - definition of, 496, 498
    - example using, 509–511
    - generics, 507–509
    - handles for, 511–514
    - method information, 503–505
    - module information, 498–500
    - properties, 507
    - tokens for, 496, 511–514

# APIs (application program interfaces) (continued)

---

## **APIs (application program interfaces) (continued)**

- type information, 500–502
- type instances, constructing, 505–506
- reflection APIs, 496–497
- Windows APIs, history of, 3–4
- XML APIs, 5

## **APM (asynchronous programming model), 385–387, 526–527**

### **AppDomain-Local Storage (ALS), 397**

### **AppDomains (application domains)**

- creating, 392
- definition of, 392
- events exposed by, 394
- isolation between, 354–355
- isolation of, 394–397
- loading code into, 393
- marshaling, 393
- unloading, 393

### **Append method, StringBuilder class, 202**

### **AppendFormat method, StringBuilder class, 202**

### **application program interfaces. See APIs**

### **APTCA (System.Security.AllowPartiallyTrustedCallers Attribute), 332–333**

### **arglist instruction, IL, 550**

### **ArgumentException exception, 210–211**

### **ArgumentNullException exception, 211**

### **ArgumentOutOfRangeException exception, 211**

### **arguments of methods, 30, 89–90**

### **arithmetic operations, IL, 91**

### **ArithmeticException exception, 210**

### **arrays**

- binary search for, 224
- cloning, 221–222
- collections interoperability, 221
- constructing, 216
- converting elements of, 224
- copying elements of, 222–223
- definition of, 215–216
- dynamic access of, 223–224
- fixed arrays, 225
- IL, 99
- jagged arrays, 218–220
- length of, 216
- multidimensional arrays, 217–220
- rank (number of dimensions), 216, 219
- rectangular arrays, 217–218
- reversing order of, 225
- single-dimensional (vectors), 216–217, 225
- sorting, 225

### **as keyword, C#, 98**

## **ASCII (American Standard Code for Information Interchange) encoding, 320**

## **ASP (Active Server Pages), 4**

## **Aspect Oriented Programming (AOP), 440**

## **ASP.NET**

- definition of, 5
- history of, 4
- memory leaks and, 408

## **AsReadOnly method, lists, 237**

## **assembly**

- bootstrapper in, 136
- definition of, 5, 133–134
- delay signing, 143
- disassembling into IL, 83
- format of, 133
- friend assemblies, 145–146
- identity of, 137–138
- info APIs for, 498–500
- loading for execution
  - binding, 146, 147–150
  - debugging load process, 149
  - domain neutrality, 152–154
  - dynamic assembly loading, 156–160
  - load contexts for, 150–151
  - loading process, 146
  - loading the CLR, 154–155
  - mapping, 146
  - probing with Fusion, 146, 148–149
  - static assembly loading, 155–156
  - type forwarding, 160–162
- metadata in
  - assembly identity, 137–138
  - definition of, 135, 136–137
  - manifest and, 138–141
  - signing assembly, 141–143
  - strong name, 141
- modules in
  - definition of, 134–135
  - info APIs for, 498–500
- platform information, 499–500
- private/public key pair for, 142
- reflection-emit assemblies, 160, 529–532
- reflection-only assembly, 499
- resources in, 143–144
- shared assemblies, 144–145
- signing, 141–143
- strong name of, 133, 135, 141–143, 339
- unloading, 151

## **assembly accessibility, 25**

## **.assembly directive, IL, 83**

**assembly manifest**, 135, 138–141

**AssemblyDef** section, assembly metadata, 136, 141

**AssemblyRef** section, assembly metadata, 136, 155

**asserts**

- in CAS, 344–346
- configuring, 462
- for tracing, 446–449

**associative arrays**. *See* dictionaries

**asterisk (\*)**, in regular expressions, 470

**asynchronous delegates**, 526–527

**asynchronous exception**, 364

**asynchronous I/O**

- I/O Completion Ports for, 273–374
- with streams, 261–264

**asynchronous programming model (APM)**, 385–387, 526–527

**atomicity of transactions**, 535

**attack vector**, 329

**attributes, custom**, 64–65, 496, 514–519

**authentication**

- HTTP, 295
- simple, 347–348
- SMTP, 297

**AV (access violation)**, 12

## B

**\b escape**, in regular expressions, 468, 469

**\b meta-character**, in regular expressions, 476

**background threads**, 365–366

**backreferencing**, in regular expressions, 478

**backward interoperability**, 423–428

**base keyword**, C#, 40

**BCL (Base Class Libraries)**

- definition of, 5
- primitive types, 17–18, 172

**Beep** method, Console class, 281

**BeginRead** method, Stream class, 261–264

**BeginReceive** method, sockets, 287

**BeginSend** method, sockets, 286

**BeginWrite** method, Stream class, 261

**beq instruction**, IL, 92, 569

**Berkeley Sockets API**. *See* sockets

**bge instruction**, IL, 92, 569

**bgt instruction**, IL, 569–570

**binary instruction size**, 86

**binary readers**, 264, 268–271

**binary resources**, 316–317

**binary search of arrays**, 224

**binary writers**, 264, 268–271

**BinarySearch** method

- arrays, 224
- lists, 237

**Bind** method, sockets, 283, 287–288

**binding an assembly**, 146, 147–150

**binding, reflection**, 497

**BindingFlags** enumeration, 502

**bindings, caching**, 513–514

**bitwise operations**, IL, 91

**ble instruction**, IL, 92, 570

**blittable types**, 434

**blt instruction**, IL, 92, 570–571

**bne instruction**, IL, 571

**BOM (byte-order-mark)**, 321

**books**. *See* publications

**bool type**, IL, 17, 172

**BooleanSwitch** class, 455

**bootstrapper in assembly**, 136

**box instruction**, IL, 93, 116, 563

**boxing**

- of collections, 73–74
- definition of, 23
- of values, 93

**br instruction**, IL, 91, 550, 571

**brackets ([ ])**, in regular expressions, 465, 470, 471

**branch instructions**, IL, 91–92

**break instruction**, IL, 551

**brfalse instruction**, IL, 91, 551, 571

**brinst instruction**, IL, 91

**brnull instruction**, IL, 91

**brtrue instruction**, IL, 91, 551, 571

**brzero instruction**, IL, 91

**buffered streams**, 260–261, 278

**BufferedStream** class, 261, 278

**builder APIs**, 529–532

**byref (pass-by-reference) arguments**, 33–35, 90

**byte-order-mark (BOM)**, 321

**byval (pass-by-value) arguments**, 33–34

## C

**\c escape**, in regular expressions, 469

**C# language**

- anonymous delegates, 63–64
- anonymous methods, 527–529
- attribute targets, qualifiers for, 518
- books about, 79, 213
- constructor chaining, 40–41
- default constructors, 39–40
- escape character for backslashes, 188
- field initialization in constructors, 41–42
- is and as keywords, 98
- iterators, 234–236
- lock blocks, 371

### **C# language (continued)**

- managed code written in, 5
- mixed mode accessibility for properties, 45
- null type, 24
- operators, list of, 48–49
- output parameters, 34
- primitive types, list of, 172
- properties, 43
- support for, 16
- type safety and, 10
- typing strategy of, 13, 14, 15

### **C++ language**

- books about, 79–80, 254
- interoperability and, 404
- primitive types, list of, 172
- support for, 16
- templates, collections and, 72, 79, 227
- type mappings, 434
- type safety and, 11
- typing strategy of, 13

### **C++/CLI language**

- conditional compilation for, 451
- generics support, 71
- interoperability and, 431
- managed code written in, 5
- operators, list of, 48–49
- type safety and, 10

### **caching bindings, 513–514**

### **calendars, regional, 308**

### **call instruction, IL, 85, 93, 127–128, 562**

### **calli instruction, IL, 93, 94, 129, 562**

### **callvirt instruction, IL, 93, 128, 563**

### **Capture class, 489–490**

### **caret (^), in regular expressions, 475**

### **CAS (code access security)**

- applying at runtime, 341–346
- asserts, 344–346
- code group, 331, 334–335
- declarative code, 342–343
- definition of, 329–330
- demands, 343–344
- denies, 344–346
- evidence, 330, 334–335
- imperative code, 342–343
- levels of trust, 332–335
- permission sets, 332, 340–341
- permissions, 331, 335–340, 341
- permits, 346
- policy, 332, 341
- protected operations, 333–334
- security context, preserving, 346
- security transparency, 346

### **CAs (custom attributes), 64–65, 496, 514–519**

### **case conversions, strings, 188, 323–325**

### **CasPol.exe utility, 341**

### **castclass instruction, IL, 97, 563**

### **catch blocks. See try/catch blocks**

### **.cctor, constructor name, IL, 42**

### **CDO (Collaboration Data Objects), 298**

### **ceq instruction, IL, 91, 551**

### **CERs (Constrained Execution Regions)**

- aggressive hosts, reliable cleanup for, 418
- critical finalization and, 418–419
- definition of, 417–418
- guaranteed cleanup of critical regions, 419–420
- inline critical regions, 419
- reliability contracts, 420–421

### **cgt instruction, IL, 91, 551**

### **chaining of constructors, 40–41**

### **change notifications, 276–278**

### **ChangeType method, 201**

### **char type, IL, 17, 172**

### **character classes, in regular expressions, 470–473**

### **CIL (Common Intermediate Language). See IL (Intermediate Language)**

### **ckfinite instruction, IL, 551**

### **class constraint, 77**

### **class constructors, 42**

### **.class directive, IL, 83**

### **class keyword, C#, 19**

### **classes (character classes), in regular expressions, 470–473**

### **classes (reference types). See reference types**

### **Clear method**

- Console class, 281
- dictionaries, 240

### **CLI (Common Language Infrastructure) specification**

- books about, 167, 213
- definition of, 9

### **client-side HTTP, 294–295**

### **client-side sockets, 290**

### **Clipboard, permission to access, 338**

### **Clone method, arrays, 221–222**

### **Close method**

- sockets, 284, 287
- Stream class, 260

### **closed (constructed) type of generics, 75**

### **CLR (Common Language Runtime)**

- books about, 78, 166, 213
- definition of, 5, 81–82
- languages supported by, 16
- loading in an assembly, 154–155
- version 2.0 improvements, 7

### **clt instruction, IL, 91, 551–552**

- code access security.** *See* **CAS**
- code group, CAS, 331, 334–335**
- coercion, 49**
- Collaboration Data Objects (CDO), 298**
- Collect method, GC class, 203**
- CollectionCount method, GC class, 203**
- collections.** *See also* **generic collections**
- definition of, 225–226
  - generics and, 72–74
  - interoperability with arrays, 221
  - weakly typed collections, 246–247
- COM (Component Object Model)**
- apartments, 390–391, 402
  - backward interoperability, 423–428
  - books about, 401, 436
  - DCOM (Distributed COM), 422–423
  - forward interoperability, 428–430
  - history of, 4
  - interoperability with, 421–430
  - visibility of, 430
- COM+, 423**
- commenting regular expressions, 474**
- Common Intermediate Language (CIL).** *See* **IL (Intermediate Language)**
- Common Language Infrastructure (CLI) specification**
- books about, 167, 213
  - definition of, 9
- Common Language Runtime.** *See* **CLR**
- Common Type System.** *See* **CTS**
- compaction, 122**
- Compare method, strings, 188, 323**
- CompareTo method, strings, 188, 323**
- Comparison<T> delegate, 251–252, 253**
- comparisons**
- custom comparers, 250–251
  - definition of, 248
  - delegate-based comparisons, 251–252, 253
  - encapsulated comparisons, 248–250
  - IL operations for, 91
  - strings, 187–188
- compilation.** *See also* **NGen technology**
- books about, 532
  - conditional compilation, 450–451
  - dynamic programming and, 495–496
  - JIT (just-in-time) compilation
    - definition of, 5, 124–126
    - method calls and, 126–131
    - 64-bit support for, 131
  - output of (metadata and IL), 82–83
- CompilationRelaxationsAttribute attribute, C#, 138**
- compiled regular expressions, 490–493**
- Component Object Model.** *See* **COM**
- compressed streams, 278–279**
- Concat method, strings, 185**
- concatenating strings, 185**
- conceptual rank of an array, 219**
- concrete classes, 51**
- concurrency.** *See also* **threads**
- optimistic concurrency, 535
  - pessimistic concurrency, 535
  - problems with
    - deadlocks, 381–382
    - race condition, 355, 369
    - starvation, 382
- concurrent collector, 122, 123**
- conditional compilation, 450–451**
- conditionals in regular expressions, 475**
- Connect method, sockets, 283, 290**
- consistency of transactions, 535**
- Console class, 280–281**
- const keyword, C#, 28**
- constant (literal) fields, 28**
- constants, IL, 88–89**
- constrained calls, 95–96**
- Constrained Execution Regions.** *See* **CERs**
- constrained. prefix, IL, 574**
- constraints, 77–78**
- constructed (closed) type of generics, 75**
- ConstructorInfo class, 505**
- constructors**
- chaining, 40–41
  - default constructors, 39–40
  - definition of, 38–39
  - field initialization in, 41–42
  - type constructors, 42
  - unhandled exceptions in, 42–43
- Contains method, strings, 192**
- contravariant delegates, 62–63, 525–526**
- conv instruction, IL, 552–555**
- conversions**
- of array elements, 224
  - case conversions, strings, 188, 323–325
  - between data types, 201
  - delegates for, 238–239, 253–254
  - from objects to strings, 179
- Convert class, 201**
- ConvertAll method, arrays, 224**
- Converter<TInput, TOutput> delegate, 238–239, 253–254**
- Copy method**
- arrays, 222–223
  - files, 275

**CopyTo method, arrays, 222–223**  
**CorHdr.h file, 136**  
**country codes, 309**  
**covariant delegates, 62–63, 525–526**  
**cpblk instruction, IL, 555**  
**cpobj instruction, IL, 563**  
**Create method, files, 273**  
**CreateInstance method, arrays, 223–224**  
**critical finalization, 124, 418–419**  
**critical regions, 388, 419–420**  
**critical sections, 368–369, 387–388**  
**cryptographically sound random numbers, 208**  
**.ctor, constructor name, IL, 38**  
**CTS (Common Type System). See also reference types (classes); type safety; value types (structures)**  
base type of, 16  
definition of, 9–10  
languages supported by, 10–11  
primitive types, 17–18  
type hierarchy for, 16–18  
verification of type safety using, 9  
**culture codes, 309**  
**CultureInfo class, 302, 309–311**  
**cultures. See also internationalization**  
changing, 313  
default, 313  
definition of, 302, 309  
enumerating through, 311  
formatting performed by, 312, 314–315  
hierarchy of, 303–304  
invariant culture, 303, 313–314  
managing, 311–313  
neutral cultures, 303, 310  
representing, 309–310  
specific cultures, 303, 310  
specifying, 310–311  
string manipulation and, 321–325  
system-wide culture, 313  
UI culture, 312–313  
**curly braces ({}), in regular expressions, 465**  
**currency**  
matching with regular expressions, 466–467  
regional variations of, 308–309  
**custom assembly binding, 149–150**  
**custom attributes (CAs), 64–65, 496, 514–519**  
**custom character classes, in regular expression, 465, 471**  
**custom collections, 241–242**  
**custom comparers, 250–251**  
**custom exceptions, 212**

**D**  
**\d class, in regular expressions, 465, 471**  
**DAcls (Discretionary ACLs), 349**  
**date types**  
definition of, 192–195  
parsing from strings, 201  
**DCOM (Distributed COM), 422–423**  
**deadlocks**  
lock blocks for, 371  
lock leveling for, 382  
with threads, 355, 381–382  
with transactions, 538–539  
**Debug class, 443–444, 449–450**  
**debugging assembly load process, 149. See also diagnostics; tracing**  
**decimals, 183–184**  
**declarative code, 342–343**  
**declarative transactions, 544–546**  
**default constructors, 39–40**  
**Deflate algorithm, 278–279**  
**delay abort regions, 364**  
**delay signing an assembly, 143**  
**delegate keyword, C#, 60**  
**delegate-based comparisons, 251–252, 253**  
**delegates**  
anonymous delegates, 63–64  
asynchronous delegates, 526–527  
contravariant delegates, 62–63, 525–526  
covariant delegates, 62–63, 525–526  
creating, 60–62  
defining, 519–521  
definition of, 60, 519  
dynamic method calls and, 129  
events and delegate chains, 523  
in-memory representation of, 521–522  
invoking, 522–523  
parameterized, 523–525  
**Delete method**  
directories, 276  
files, 275  
**demands, CAS, 343–344**  
**denies, CAS, 344–346**  
**dependent transactions, 543–544**  
**diagnostics. See also tracing**  
MDAs (Managed Debugging Assistants), 439  
performance counters, 439  
Windows Event Log, 338, 439, 452  
**dictionaries**  
always-sorted dictionaries, 241  
definition of, 231–233  
standard dictionaries, 239–240

**Dictionary<TKey, TValue> class, 239–240**

**dimensions of an array, 216**

**directories**

- change notifications for, 276–278
- copying, 275
- creating, 274
- deleting, 275–276
- moving, 275

**DirectoryInfo class, 274**

**Discretionary ACLs (DACLs), 349**

**Dispose method**

- Object type, 177–179
- sockets, 287
- Stream class, 260
- transactions and, 540

**Distributed COM (DCOM), 422–423**

**distributed transactions, permission to use, 339**

**div instruction, IL, 91, 555**

**DNS server, external, permission for, 338**

**domain neutrality, 152–154**

**dot (.) class, in regular expressions, 471**

**dup instruction, IL, 555**

**durability of transactions, 535**

**durations, 195**

**dynamic array access, 223–224**

**dynamic assembly loading, 156–160**

**dynamic method calls, 129**

**dynamic programming, 495–496. See also reflection**

**dynamic typing, 13–16**

## E

**\e escape, in regular expressions, 468**

**early binding, 495**

**eldest generation, 117**

**e-mail address, matching, 466**

**e-mail (SMTP) protocol, 297–298**

**embedded resources in assembly, 143–144**

**/embedresource switch, C#, 144**

**emitting code and metadata, 529–532**

**encapsulated comparisons, 248–250**

**encodings**

- books about, 325
- internationalization and, 302, 320–321

**end of stream (EOS) condition, 256–257**

**endfilter instruction, IL, 555**

**endfinally instruction, IL, 555**

**EndRead method, Stream class, 261, 264**

**EndReceive method, sockets, 287**

**EndSend method, sockets, 286**

**EndWrite method, Stream class, 261**

**Enterprise policy level, 341**

**Enterprise Services (ES), transaction management using, 544–546**

**.entrypoint directive, IL, 83, 136**

**enum keyword, C#, 66**

**enumerations**

- definition of, 65–67
- flags-style enumerations, 67–68, 69
- helper methods for, 69
- type safety and, 68–69

**enumerators, 233–236**

**environment variables, permission to access, 337**

**EnvironmentPermission class, 337**

**EOS (end of stream) condition, 256–257**

**ephemeral generations, 117**

**epilogue of method, 129–130**

**equality, testing for. See comparisons**

**Equals method**

- Object type, 173–176
- strings, 187–188

**ES (Enterprise Services), transaction management using, 544–546**

**event keyword, C#, 45–46**

**EventLogPermission class, 338**

**events**

- definition of, 45–46, 382–383
- delegate chains and, 523
- exposed by AppDomains, 394
- monitor-based events, 383–384
- timers, 385
- Win32 events, 384–385

**evidence, CAS, 330, 334–335**

**exceptions**

- catching exceptions, 100
- class hierarchy for, 110–111
- compared to tracing, 441
- custom exceptions, 212
- definition of, 99–100
- fail fast, 111
- fault blocks, 105–106
- finally blocks
  - definition of, 106–107
  - ensuring memory release using, 409
  - two-pass exceptions and, 112–113
  - unhandled exceptions and, 109
- list of exceptions, 208–212
- in methods, 38
- performance of, 113–115
- rethrowing exceptions, 105
- SEH (Structured Exception Handling), 101
- statistics regarding, 114
- throwing exceptions, 99, 100, 101–102
- throwing non-exception objects, 107–108

### exceptions (continued)

- `try/catch` blocks for
  - catch on Boolean filter, 104–105
  - catch on type filter, 102–104
  - definition of, 102
  - ensuring memory release using, 409
  - examples of, 100–101
- two-pass exception model for, 101, 111–113
- undeniable exceptions, 109–110
- unhandled exceptions
  - in constructors, 42–43
  - definition of, 99, 108–109
- wrapped exceptions, 108

### ExecutionEngineException exception, 210

### extends keyword, IL, 49–50

### Extensible Markup Language (XML)

- APIs, 5
- parsing with regular expressions, 480–481
- for resource files, 316–317

## F

### `\f` escape, in regular expressions, 468

### F# language

- downloading, 16
- type safety and, 11
- typing strategy of, 13, 14, 15

### fail fast, 111

### family (protected) accessibility, 25

### fastcall calling convention, 127

### fault blocks, 105–106

### fibers, threads mapped to, 355

### field initialization in constructors, 41–42

### FieldDef section, assembly metadata, 136

### fields

- constant (literal) fields, 28
- definition of, 26–27
- memory layout for structs, controlling, 29–30
- read-only fields, 27–28
- size of, 27

### FIFO queues, 244–245

### file ACLs, 350–351

### File class, opening files using, 271–272

### FileInfo class, opening files using, 271

### FileIOPermission class, 336–337

### files

- access control for, 273, 348–351
- change notifications for, 276–278
- copying, 275
- creating, 273
- deleting, 275–276

- file system management, 274–278
  - moving, 275
  - opening, 271–273
  - permission to access, 336–337
  - temporary files, 276

### FileStream class

- file handles in, 271
- reading and writing from, 273

### FileSystemWatcher class, 276

### finalization

- critical finalization, 124, 418–419
- definition of, 123, 203, 410–411
- finalizers for objects, 123–124, 177–179
- when `finally` blocks don't execute, 107

### Finalize method, Object type, 123–124, 177

### finally blocks

- definition of, 106–107
- ensuring memory release using, 409
- two-pass exceptions and, 112–113
- unhandled exceptions and, 109

### first-in, first-out data structure (FIFO queues), 244–245

### FirstMatchCodeGroup class, 334

### fixed arrays, 225

### flags-style enumerations, 67–68, 69

### float32 type, IL, 18, 172

### float64 type, IL, 18, 172

### floating point types, 183–184

### Flush method, Stream class, 261

### ForEach method, lists, 237

### ForegroundColor method, Console class, 281

### fork/join patterns, 363

### formatting

- regional, 307–309
- strings, 186, 196–200

### forward interoperability, 428–430

### fragmentation, 115, 122

### friend assemblies, 145–146

### fully trusted code, CAS, 332

### functional delegate types, 238–239, 252–254

### Fusion, probing using, 146, 148–149

## G

### `\G` meta-character, in regular expressions, 476

### GAC (Global Assembly Cache), 144–145

### GacIdentityPermission class, 339

### gacutil.exe tool, 145

### garbage collection (GC)

- books about, 131
- compaction, 122
- fragmentation, 115, 122

GC class for, 202–204  
 handles, marshaling using, 435–436  
 notifying of resource consumption, 416–417  
 process of, 121  
 server garbage collection, 123  
 when occurring, 120–121  
 workstation garbage collection, 122

**GC (Garbage Collected) heap, reference types managed by, 18, 19**

**generations in small object heap, 117**

**generic collections**  
 always-sorted dictionaries, 241  
 base interfaces for, hierarchy of, 228  
 books about, 254  
 custom collections, 241–242  
 definition of, 226  
 dictionaries  
   always-sorted dictionaries, 241  
   definition of, 231–233  
   standard dictionaries, 239–240  
 enumerators, 233–236  
 FIFO queues, 244–245  
 indexable collections, 230–231  
 LIFO stacks, 243–244  
 linked lists, 245–246  
 read-only collections, 243  
 reasons to use, 226–227  
 simple collections, 229–230  
 standard dictionaries, 239–240  
 standard lists, 236–239

**generics**  
 books about, 79  
 collections and, 72–74, 226  
 constructed (closed) type of, 75  
 definition of, 69–70  
 info APIs for, 507–509  
 instantiation of, 70–72  
 open type of, 75  
 performance of, 76  
 type storage for, 75–76  
 usability and maintainability of, 76  
 version 2.0 improvements for, 7

**GetAccessControl method, files, 273**

**GetCultureInfo method, CultureInfo class, 310–311**

**GetCultures method, CultureInfo class, 311**

**GetCustomAttribute method, 519**

**GetGeneration method, GC class, 203**

**GetHashCode method, Object type, 176**

**GetLength method, arrays, 218, 224**

**GetLowerBound method, arrays, 224**

**GetTempFileName method, Path class, 276**

**GetTempPath method, Path class, 276**

**getters, 43–44, 45**

**GetTotalMemory method, GC class, 202**

**GetType method, Object type, 179**

**GetUpperBound method, arrays, 224**

**GetValue method, arrays, 224**

**Global Assembly Cache (GAC), 144–145**

**globalization, 302, 305. See also internationalization**

**greediness, in regular expressions, 479–481**

**Group class, 489–490**

**grouping, in regular expressions, 477–479**

**GZIP algorithm, 278–279**

## H

**handle recycling attack, 406–408**

**HandleCollector type, 417**

**hard binding, NGen, 164–165**

**hash codes, 176**

**hash tables, 240. See also dictionaries**

**Haskell language**  
 books about, 80  
 tail calls and, 94  
 typing strategy of, 13

**hosts, 5**

**HT (Hyper-Threading) technology, 391–392**

**HTTP (HyperText Transfer Protocol)**  
 client-side HTTP, 294–295  
 definition of, 293–294  
 listening for connections, 295  
 permission to use, 339  
 processing requests, 296–297

**HttpListener class, 295**

**HttpListenerContext class, 296–297**

**HttpListenerResponse class, 296–297**

**Hyper-Threading (HT) technology, 391–392**

## I

**ICollection<T> interface, 229–230, 247**

**IComparable<T> interface, 248–250**

**IComparer<T> interface, 250–251**

**IConvertible interface, 201**

**IDictionary<TKey, TValue> interface, 231–233, 239, 241, 247**

**IDisposable interface, 411–412**

**i18n. See internationalization**

**IEnumerable<T> interface, 233–236, 247**

**IEnumerator<T> interface, 233–236, 247**

**IEqualityComparer<T> interface, 250–251**

**IEquatable<T> interface, 248–250**

**IFormatProvider interface, 314**

## **IHashCodeProvider interface, 247**

## **IIS (Internet Information Services), 5, 294**

## **IL (Intermediate Language)**

- allocating and initializing types, 92–93
- arithmetic operations, 91
- arrays, 99
- assembling, 83
- binary instruction size for, 86
- bitwise operations, 91
- boxing and unboxing values, 93
- branch instructions, 91–92
- comparison operations, 91
- definition of, 5, 82
- disassembling, 83
- generated by compilation, 82–83
- labels, 91
- loading values on stack
  - arguments and locals, 89–90
  - arrays, 99
  - constants, 88–89
  - definition of, 87–88
  - fields, 90
  - indirect loads, 90
- macros, 568–574
- methods, calling and returning from, 93–97
- object model instructions, 562–568
- prefixes for instructions, 574–575
- primitive instructions, 550–562
- primitive types, 17–18, 171–172
- stack-based nature of, 84–86
- storing values from stack
  - arguments of methods, 89–90
  - arrays, 99
  - definition of, 87–88
  - fields, 90
  - indirect stores, 90
- type identity checks, 97–98
- type verification for, 86

## **ilasm.exe tool, 83**

## **ildasm.exe tool, 83**

## **ICollection<T> interface, 230–231, 236, 247**

## **imperative code, 342–343**

## **impersonation, 348**

## **implementation inheritance, 50**

## **indexable collections, 230–231**

## **indexer properties, 44–45**

## **IndexOf method**

- lists, 237
- strings, 192

## **IndexOutOfRangeException exception, 210**

## **indirect calls, 94, 129**

## **info APIs**

- assembly information, 498–500
- definition of, 496, 498
- example using, 509–511
- generics, 507–509
- handles for, 511–514
- method information, 503–505
- module information, 498–500
- properties, 507
- tokens for, 496, 511–514
- type information, 500–502
- type instances, constructing, 505–506

## **inheritance**

- implementation inheritance, 50
- interface inheritance, 51–57
- multiple inheritance, 54
- private interface inheritance, 54–55
- subclassing using, 50–51

## **inheritance demands, CAS, 344**

## **initblk instruction, IL, 555**

## **initobj instruction, IL, 92–93, 563**

## **initonly keyword, IL, 27**

## **input and output. See I/O**

## **Insert method, StringBuilder class, 202**

## **instance members, 26**

## **instance methods, 31**

## **instantiation. See constructors**

## **instruction reordering, 388–389**

## **int8 type, IL, 17, 172**

## **int16 type, IL, 17, 172**

## **int32 type, IL, 17, 172**

## **int64 type, IL, 17, 172**

## **integers, 180–182**

## **interface inheritance**

- abstract classes and, 51–52, 56–57
- definition of, 51
- interfaces and, 52–57
- multiple inheritance, 54
- private interface inheritance, 54–55

## **interfaces**

- definition of, 52–53
- method invocations and, 53
- reimplementation of, 55–56
- when to use, 56–57

## **Interlocked class, 378–381**

## **interlocked operations, 378–381**

## **Intermediate Language. See IL**

## **internal keyword, C#, 25**

## **InternalsVisibleToAttribute attribute, 145**

**internationalization. See also cultures; resources for cultural preferences**

books about, 325  
 definition of, 301–302  
 encodings and, 302, 320–321, 325  
 example scenarios of, 306–309  
 globalization, 302, 305  
 locale-specific features, 305  
 localization, 302, 306–307  
 process of, 305–306  
 reasons for, 304–305  
 regional formatting, 307–309  
 support for, 302  
 translation of content, 305, 306–307

**Internet Information Services (IIS), 5, 294****interoperability**

books about, 436  
 C++/CLI language for, 431  
 CERs (Constrained Execution Regions), 417–421  
 COM interoperability, 421–430  
 definition of, 404–405  
 memory management, 408–412  
 notifying GC of resource consumption, 416–417  
 P/Invoke for, 431–433  
 pointers and, 405–408  
 reasons for, 403–404  
 resource management, 410–416  
 type systems, bridging, 434–436

**InvalidAddressException exception, 210****InvalidCastException exception, 209****InvalidOperationException exception, 211****invariant culture, 303, 313–314****I/O completion ports, 273–274****I/O (input and output). See also files; streams**

books about, 298–299  
 definition of, 255–256  
 isolated storage, 337  
 overlapped I/O, 273–274  
 standard devices, 280–282

**IPEndPoint class, 290****IPrinciple interface, 347****IronPython language, managed code written in, 5****is keyword, C#, 98****IsDefined method, System.Enum type, 69****IsInRole method, IPrinciple interface, 347****isinst instruction, IL, 97–98, 563****IsMatch method, Regex class, 483–484, 488****isolated storage, 337****isolation levels**

for threads, 354–355  
 for transactions, 539

**isolation of transactions, 535****ISO-639 standard, 309****ISO-3166 standard, 309****iterators, 234–236****J****jagged arrays, 218–220****Java language**

books about, 401  
 generics and, 76  
 type safety and, 11  
 typing strategy of, 13

**Java platform, 4****JIT (just-in-time) compilation**

definition of, 5, 124–126  
 method calls and, 126–131  
 64-bit support for, 131

**jmp instruction, IL, 556****Join method, strings, 192****K****KeyContainerPermission class, 338****L****labels, IL, 91****language codes, 309****languages. See cultures; programming languages****large object GC heap, 115****last-in, first-out data structure (LIFO stacks), 243–244****LastIndexOf method**

lists, 237  
 strings, 192

**late binding, 495****LayoutKind enumeration, 29****lazy compilation of regular expressions, 490–491****lazy quantifiers, in regular expressions, 481****LCG (lightweight code generation), 532****ldarg instruction, IL, 89, 556, 571–572****ldarga instruction, IL, 90, 556****ldarga.s instruction, IL, 90****ldarg.s instruction, IL, 572****ldc instruction, IL, 88–89, 556, 572–573****ldelem instruction, IL, 99, 563–565****ldfld instruction, IL, 90, 566****ldflda instruction, IL, 90, 566****ldftn instruction, IL, 556****ldind instruction, IL, 90, 556–557****ldlen instruction, IL, 99, 566****ldloc instruction, IL, 89, 557, 573**

**ldloca instruction, IL, 90, 558, 573**  
**ldloca.s instruction, IL, 90**  
**ldnull instruction, IL, 89, 558**  
**ldobj instruction, IL, 566**  
**ldsfd instruction, IL, 90, 566**  
**ldsfla instruction, IL, 90, 566**  
**ldstr instruction, IL, 88, 566**  
**ldtoken instruction, IL, 558**  
**ldvirtfn instruction, IL, 558**  
**leave instruction, IL, 558**  
**ledlema instruction, IL, 99**  
**levels of trust, CAS, 332–335**  
**LIFO stacks, 243–244**  
**lightweight code generation (LCG), 532**  
**limping along, 440**  
**link demands, CAS, 343**  
**linked lists, 245–246**  
**linked resources in assembly, 143–144**  
**LinkedList<T> class, 245–246**  
**/linkresource switch, C#, 144**  
**LISP language**  
  books about, 80  
  tail calls and, 94  
  typing strategy of, 13  
**Listen method, sockets, 283, 288**  
**lists**  
  ACLs (Access Control Lists), 348–351, 374  
  linked lists, 245–246  
  standard lists, 236–239  
**literal (constant) fields, 28**  
**literals, in regular expressions, 464, 468–469**  
**load acquire, instruction reordering, 388**  
**load contexts, 150–151**  
**loading an assembly**  
  binding, 146, 147–150  
  debugging load process, 149  
  domain neutrality, 152–154  
  dynamic assembly loading, 156–160  
  load contexts for, 150–151  
  loading process, 146  
  loading the CLR, 154–155  
  mapping, 146  
  probing with Fusion, 146, 148–149  
  static assembly loading, 155–156  
  type forwarding, 160–162  
**locality of reference, 117–118**  
**localization. See also internationalization**  
  books about, 325  
  of content, 306–307  
  definition of, 302  
**localloc instruction, IL, 558**

**locals**  
  definition of, 31–32  
  loading and storing from stack, 89–90  
**lock blocks, C#, 371**  
**lock leveling, 382**  
**Lock method, files, 273**  
**locks. See also deadlocks**  
  for critical sections, 368–369  
  for files, 273  
  spin-locks, 379–380  
**logical execution stack, 84**  
**lookahead meta-characters, 476–477**  
**lookbehind meta-characters, 476–477**

## M

**Machine policy level, 341**  
**macros, IL, 568–574**  
**managed code. See also interoperability**  
  definition of, 81  
  generating type libraries from, 429–430  
  proxies for, 424–426  
**Managed Debugging Assistants (MDAs), 439**  
**managed pointers, 35**  
**managed threads, 355**  
**manifest, assembly, 135, 138–141**  
**mapping an assembly, 146**  
**maps. See dictionaries**  
**marker attributes, 516**  
**marshaling, 434, 435–436**  
**Match class, 488–489**  
**Match method, Regex class, 484, 488**  
**Matches method, Regex class, 484, 488**  
**.maxstack directive, IL, 83**  
**MDAs (Managed Debugging Assistants), 439**  
**MemberRef section, assembly metadata, 136**  
**members of types. See type members**  
**memory consistency model, 388**  
**memory corruption, type safety and, 11–12**  
**memory fence, instruction reordering, 389**  
**memory gates, 120**  
**memory layout**  
  for delegates, 521–522  
  for objects, 20–21  
  for structs, controlling, 29–30  
  for values, 21–22  
**memory management**  
  allocation of memory, 115–120  
  definition of, 115  
  finalization, 123–124  
  garbage collection (GC), 120–123

- interoperability and, 408–412
- Out of Memory (OOM), 119–120
- Stack Overflow (SO), 118–119
- memory models**
  - books about, 402
  - definition of, 388–390
- memory pressure, 203–204, 416–417**
- memory streams, 279–280**
- MemoryStream class, 279–280**
- merging strings, 192**
- meta-characters, in regular expressions**
  - alternations, 474–475
  - character classes, 473
  - comments, 474
  - conditionals, 475
  - definition of, 464, 469–470
  - greediness and, 479–481
  - grouping, 477–479
  - lookahead and lookbehind meta-characters, 476–477
  - positional matching, 464, 475–477
  - quantifiers, 464, 470, 481
- metadata**
  - in assembly
    - assembly identity, 137–138
    - definition of, 135, 136–137
    - manifest and, 138–141
    - signing assembly, 141–143
    - strong name, 141
  - emitting, 529–532
  - generated by compilation, 82–83
- .method directive, IL, 83**
- method table, 21, 125**
- MethodDef section, assembly metadata, 136, 141**
- MethodInfo type, 503**
- MethodRef section, assembly metadata, 136, 141**
- methods**
  - abstract methods, 51
  - anonymous methods, 527–529
  - arguments of
    - definition of, 30
    - loading and storing from stack, 89–90
    - passing style of, 33–35
  - building, 530–531
  - calling, 93–97, 126–131
  - constrained calls, 95–96
  - definition of, 30–31
  - epilogue for, 129–130
  - exception handlers in, 38
  - indirect calls, 94, 129
  - info APIs for, 503–505
  - instance methods, 31
  - of interfaces, invocation of, 53
  - invoking, 504
  - locals of
    - definition of, 31–32
    - loading and storing from stack, 89–90
  - new slots, 37–38
  - output parameters, 34
  - overloading, 32
  - overriding, 36–37
  - parameters of, 30
  - private interface inheritance and, 55
  - prologue for, 129–130
  - return parameter of, 30
  - returning from, 94
  - sealing, 57
  - static calls, 93–94
  - static methods, 31
  - subclassing and, 35–38
  - tail calls, 94–95
  - variable argument methods, 35
  - virtual calls, 93–94
  - virtual methods
    - calling, 93–94, 128
    - definition of, 36–37
    - nonvirtual calls to, 96–97
- Microsoft Transaction Server (MTS), 423**
- Microsoft Windows APIs, history of, 3–4**
- MIME (Multipurpose Internet Mail Extensions), 297**
- missed pulse, 384**
- MissingFieldException exception, 210**
- MissingMethodException exception, 210**
- mixed mode accessibility for properties, 45**
- mkrefany instruction, IL, 558**
- ML language**
  - tail calls and, 94
  - typing strategy of, 13
- MMC, modifying policy using, 341**
- Module class, 498**
- modules in assembly**
  - definition of, 134–135
  - info APIs for, 498–500
- money. See currency**
- monitor-based events, 383–384**
- monitors for objects, 369–372**
- Move method, files, 275**
- MoveTo method, directories, 275**
- mscorlib.dll file, 136**
- MTA (Multi-Threaded Apartment), 390**
- MTS (Microsoft Transaction Server), 423**
- mul instruction, IL, 91, 559**
- multidimensional arrays, 217–220**

**multiple inheritance, 54**  
**Multipurpose Internet Mail Extensions (MIME), 297**  
**mutexes, 372–373**

## N

**\n escape, in regular expressions, 468**

**NA (Neutral Apartment), 390**

**NamedPermissionSet class, 340**

**namespaces**

aliasing, 59

defining, 58–59

definition of, 58

resolving references to, 59–60

**native int type, IL, 18, 172, 405**

**native unsigned int type, IL, 172**

**NCL (Networking Class Libraries), 282**

**neg instruction, IL, 559**

**negative infinity, 183**

**nested type definitions, 26**

**.NET Framework**

books about, 78, 212–213

components of, 5–6

history of, 4–5

version 2.0 improvements, 7

**networking. See also sockets**

books about, 299

definition of, 282

information about, obtaining, 290

OSI model for, 291

protocols

definition of, 291

HTTP, 293–297, 339

SMTP (e-mail), 297–298

TCP/IP, 291–292

UDP, 292–293

**Networking Class Libraries (NCL), 282**

**NetworkInterface type, 290**

**NetworkStream class, 286–287**

**Neutral Apartment (NA), 390**

**neutral cultures, 303, 310**

**new constraint, 78**

**new keyword, C#, 37, 116**

**new slots, 37–38**

**newarr instruction, IL, 99, 116, 216, 566**

**newobj instruction, IL, 92, 116, 566**

**NGen technology**

base addresses, 163–164

benefits of, 165–166

books about, 166, 167

definition of, 82, 125–126, 162–163

disadvantages of, 166

hard binding, 164–165

string freezing, 165

using, 163

**ngen.exe tool, 163**

**nonvirtual calls to virtual methods, 96–97**

**nop instruction, IL, 559**

**not instruction, IL, 91, 559**

**NotImplementedException exception, 211–212**

**NotSupportedException exception, 211–212**

**null unification, 23–24**

**null value**

objects using, 19

values not using, 22

**nullable value type, 93**

**NullReferenceException exception, 209**

**numeric primitives, 180–184**

## O

**\0 escape, in regular expressions, 469**

**object model instructions, IL, 562–568**

**object type, IL**

converting to string, 179

definition of, 172

equality methods for, 173–176

finalizers for, 177–179

hash codes for, 176

type identity of, 179

**ObjectDisposedException exception, 211**

**objects**

definition of, 18

memory layout for, 20–21

unification with values, 22–24

**OLE Automation, 4**

**OleView.exe utility, 423**

**OOM (Out of Memory), 119–120**

**Open methods, files, 271–273**

**Open Systems Interconnection (OSI) model, 291**

**open type of generics, 75**

**OpenRead method, files, 273**

**OpenWrite method, files, 273**

**operating systems, books about, 132**

**operator overloading, 47–49**

**optimistic concurrency, 535**

**or instruction, IL, 91, 559**

**OSI (Open Systems Interconnection) model, 291**

**out keyword, C#, 34, 35**

**Out of Memory (OOM), 119–120**

**OutOfMemoryException exception, 209**

**overflow, floating points, 183**

**overlapped I/O, 273–274**

**overloading methods, 32**

overloading operators, 47–49  
 override keyword, C#, 36  
 overriding methods, 36–37

## P

**padding strings, 190–191**  
**PadLeft method, strings, 190**  
**PadRight method, strings, 190**  
**parallelism. See threads**  
**ParamDef section, assembly metadata, 136**  
**parameterized delegates, 523–525**  
**parameters of methods, 30**  
**parametric polymorphism, 69, 79**  
**parentheses (()), in regular expressions, 477–479**  
**Parse method**  
   cultures and, 321–322  
   definition of, 200–201  
   exceptions and, 114–115  
**ParseExact method, strings, 201**  
**parsing**  
   strings  
     cultures and, 321–322  
     definition of, 200–201  
     exceptions and, 114–115  
     XML, 480–481  
**partial trust, unverifiable code and, 12**  
**partially trusted code, CAS, 332**  
**pass by copy. See pass-by-value (byval) arguments**  
**pass-by-reference (byref) arguments, 33–35, 90**  
**pass-by-value (byval) arguments, 33–34**  
**Path class, 274–275**  
**patterns. See regular expressions**  
**PEB (Process Environment Block), 347, 354**  
**PE/COFF (Portable Executable/Common Object File Format), 133**  
**performance**  
   exception handling and, 113–115  
   generics and, 76  
   performance counters, 338, 439  
   threading and, 353  
**PerformanceCounterPermission class, 338**  
**permission sets, CAS, 332, 340–341**  
**permissions, CAS, 331, 335–340, 341**  
**permits, CAS, 346**  
**pessimistic concurrency, 535**  
**peverify.exe tool, 86**  
**physical stack, 84**  
**PIA (Primary Interop Assembly), 425**  
**pinning, 436**

**P/Invoke (Platform Invoke), 431–433**  
**pipe character (|), in regular expressions, 474**  
**platform information in assembly, 499–500**  
**plus symbol (+), in regular expressions, 470**  
**pointer type, 192**  
**pointers, interoperability and, 405–408**  
**policy, CAS, 332, 341**  
**polymorphism**  
   parametric polymorphism, 69, 79  
   subclassing and, 49–50  
**pop instruction, IL, 559**  
**Portable Executable/Common Object File Format (PE/COFF), 133**  
**positional matching, in regular expressions, 464, 475–477**  
**positive infinity, 183**  
**precompilation of regular expressions, 491**  
**predicate functions, 254**  
**Predicate<T> delegate, 239, 254**  
**prefixes for instructions, IL, 574–575**  
**Primary Interop Assembly (PIA), 425**  
**primary module in assembly, 135**  
**primitive instructions, IL, 550–562**  
**primitive types**  
   boolean, 184  
   conversions between, 201  
   dates and times, 192–195  
   definition of, 171–172  
   hierarchy of, 17–18  
   list of, 172  
   numbers, 180–184  
   objects, 173–179  
   parsing from strings, 200–201  
   pointer, 192  
   strings, 184–192  
**PrincipalPermission class, 338**  
**priority inversion, 382**  
**private accessibility, 25**  
**private interface inheritance, 54–55**  
**private keyword, C#, 25**  
**private/public key pair for assembly, 142**  
**probing with Fusion, 146, 148–149**  
**Process Environment Block (PEB), 347, 354**  
**processes**  
   creating, 400  
   definition of, 397  
   existing, accessing, 397–399  
   interacting with, 399–400  
   isolation between, 354–355  
   terminating, 400–401

## **Program IL, assembly metadata, 136**

### **programming. See also managed code**

- dynamic, 495–496
- on Java platform, history of, 4
- unmanaged code, 403
- on Windows platform, history of, 3–4

### **programming language design, books about, 78–79**

### **programming languages**

- CLR support for, 16
- compilation of, 82
- CTS support for, 10–11

### **prologue of method, 129–130**

### **properties**

- definition of, 43–44
- indexer properties, 44–45
- info APIs for, 507
- mixed mode accessibility for, 45

### **protected (family) accessibility, 25**

### **protected internal keywords, C#, 25**

### **protected keyword, C#, 25**

### **protected operations, CAS, 333–334**

### **protected regions, 103**

### **protocols**

- definition of, 291
- HTTP, 293–297, 339
- SMTP (e-mail), 297–298
- TCP/IP, 291–292
- UDP, 292–293

### **pseudo-random numbers, 207**

### **public accessibility, 25**

### **public key for assembly, 142**

### **public keyword, C#, 25**

### **publications**

- about C# language, 79, 213
- about C++ language, 79–80
- about CLI, 167, 213
- about CLR, 78, 166, 213
- about COM, 401
- about COM apartments, 402
- about generic collections, 254
- about generics, 79
- about internationalization, 325
- about interoperability, 436
- about I/O, 298–299
- about LISP language, 80
- about memory models, 402
- about .NET Framework, 78, 212–213
- about networking, 299
- about NGen technology, 166, 167
- about operating systems, 132
- about parametric polymorphism, 79

- about programming, 254
- about programming language design, 78–79
- about Python language, 80
- about reflection, 532
- about regular expressions, 493
- about Scheme language, 80
- about security, 351
- about strings, 213
- about threading and concurrency, 401–402
- about tracing, 462
- about transactions, 548
- about type systems, 78–79
- about VB language, 79
- about virtualizing architectures, 131–132

### **PublicKey section, assembly manifest, 142**

### **PublisherIdentityPermission class, 339**

### **Python language**

- books about, 80
- downloading, 16
- type safety and, 10
- typing strategy of, 14, 15

## **Q**

### **quantifiers, in regular expressions, 464, 470, 481**

### **question mark (?), in regular expressions, 474, 475**

### **Queue<T> class, 244–245**

### **queues, FIFO, 244–245**

## **R**

### **\r escape, in regular expressions, 468**

### **race condition, 355, 369**

### **random number generation, 207–208**

### **rank (number of dimensions) of an array, 216, 219**

### **Read method, Stream class, 257–258**

### **ReadByte method, Stream class, 257**

### **readers, 255–256**

### **reader-writer locks, 374–378**

### **ReaderWriterLock class, 374–378**

### **ReadLine method, strings, 266**

### **read-only collections, 243**

### **read-only fields, 27–28**

### **readonly keyword, C#, 27**

### **readonly. prefix, IL, 574**

### **ReadOnlyCollection<T> class, 243**

### **ReadT methods, binary data, 269–270**

### **ReadToEnd method, strings, 266**

### **Receive method, sockets, 284, 287**

### **rectangular arrays, 217–218**

### **recursion, tail calls for, 94–95**

### **ref keyword, C#, 33, 35**

**refanytype instruction, IL, 559**

**refanyval instruction, IL, 559**

**reference types (classes). See also objects**

- abstract classes, 51–52, 56–57
- allocating and initializing, 92–93
- compared to value types, 18
- concrete classes, 51
- creating, 19
- definition of, 16, 18–19
- weak references, 204–205
- when to use, 18–19

**ReferenceEquals method, Object type, 176**

**reflection**

- activation, 496
- anonymous methods, 527–529
- APIs for, list of, 496–497
- binding, 497
- books about, 532
- caching bindings, 513–514
- custom attributes (CAs), 496, 514–519
- definition of, 14, 495–496
- delegates, 496
- handles, 496, 511–514
- info APIs, 496
- permission for, 337
- tokens, 496, 511–514

**reflection-emit assemblies, 160, 529–532**

**reflection-only assembly, 499**

**ReflectionPermission class, 337**

**regional formatting, 307–309**

**register-based machines, 85–86**

**registry, permission to access, 337**

**RegistryPermission class, 337**

**regular expressions. See also meta-characters, in regular expressions**

- alternations, 474–475
- backreferencing in, 478
- books about, 493
- Capture class, 489–490
- character classes in, 470–473
- commenting, 474
- compiled expressions, 490–493
- conditionals in, 475
- custom character classes in, 465
- definition of, 463
- examples of, 465–467, 480–481
- expression syntax, 464
- greediness and, 479–481
- Group class, 489–490
- grouping, 477–479
- literals in, 464, 468–469

Match class, 488–489

positional matching, 464, 475–477

quantifiers in, 464, 470, 481

Regex class, 482–488

wildcards in, 464

**Relative Virtual Address (RVA) statics, 19**

**reliability contracts, 420–421**

**reliability, version 2.0 improvements for, 7**

**rem instruction, IL, 91, 560**

**Remove method**

StringBuilder class, 202

strings, 189

**RemoveMemoryPressure method, GC class, 203–204**

**rendezvousing, 386**

**Replace method**

Regex class, 485–487, 488

StringBuilder class, 202

strings, 189

**ResGen.exe tool, 315–317**

**resource management**

garbage collection and, 416–417

interoperability and, 410–416

**resource manager (RM), 533–534, 546**

**Resource Manifests, assembly metadata, 136**

**.resources files, 315–316**

**resources for cultural preferences**

accessing, 318–319

binary resources, 316–317

books about, 325

creating, 315–317

definition of, 302, 315

packaging and deploying, 317–318

strongly typed resources, 318–319

text resources, 315–316

using, 307

weakly typed resources, 319

**resources in assembly, 143–144**

**resources (information). See publications**

**resurrection, 124**

**.resx files, 316–317**

**ret instruction, IL, 560**

**rethrow instruction, IL, 105, 567**

**return parameter of method, 30**

**Reverse method**

arrays, 225

lists, 237

**reversing arrays, 225**

**reversing lists, 237**

**RFC 2279, 320**

**RM (resource manager), 533–534, 546**

**Running state, Thread object, 358, 361**

runtime constraints, 77–78  
runtime type checking, 58  
RuntimeWrappedException exception, 210  
RVA (Relative Virtual Address) statics, 19

## S

\s class, in regular expressions, 472  
SACLs (System ACLs), 349  
sandboxing mechanism, 329–330. *See also* CAS (code access security)  
satellite assemblies, 317–318  
scalars (integers), 180–182  
Scheme language  
  books about, 80  
  downloading, 16  
  typing strategy of, 14, 15  
sealed keyword, C#, 57  
sealed methods, 57  
sealed types, 57  
sealed, value types as, 19  
searching arrays, 224  
searching lists, 237  
searching strings, 192  
Secure Socket Layer (SSL), 293  
security. *See also* CAS (code access security)  
  books about, 351  
  handle recycling attack, 406–408  
  importance of, 329–330  
  user-based security  
    access controls, 348–351  
    impersonation, 348  
    simple authentication, 347–348  
    user identity for, 347–348  
security context, preserving, CAS, 346  
security transparency, CAS, 346  
SecurityException exception, 210  
SecurityPermission class, 336  
Seek method, Stream class, 259–260  
SEH (Structured Exception Handling), 101  
semaphores, 373–374  
Send method, sockets, 283, 286  
separator characters, regional, 308  
sequential consistency, 388  
serial port, communication using, 282  
server garbage collection, 123  
server-side sockets, 287–289  
SetAccessControl method, files, 273  
SetBufferSize method, Console class, 281  
SetCursorPosition method, Console class, 281  
setters, 43–44, 45  
SetValue method, arrays, 224  
shared assemblies, 144–145  
shl instruction, IL, 91, 560  
shr instruction, IL, 91, 560  
signaled mutexes, 372  
Signature Table, assembly metadata, 136  
signatures, P/Invoke, 431–433  
signing an assembly, 141–143  
simple authentication, 347–348  
simple collections, 229–230  
Simple Mail Transfer Protocol (SMTP), 297–298  
Single Threaded Apartment (STA), 390  
single-dimensional arrays (vectors), 216–217, 225  
SitIdentityPermission class, 339  
64-bit architecture  
  JIT (just-in-time) compilation supporting, 131  
  version 2.0 improvements for, 7  
sizeof instruction, IL, 22, 560  
sizeof(T) operator, C#, 22  
small object GC heap  
  generations in, 117  
  memory management of, 115  
SMTP server, permission to access, 338  
SMTP (Simple Mail Transport Protocol), 297–298  
SmtplibClient class, 297  
SmtplibMail class, 298  
sn.exe tool, 142, 143  
SO (Stack Overflow), 118–119  
Social Security Number (SSN), matching, 465–466  
Socket method, sockets, 283  
sockets  
  accepting requests, 283, 288–289  
  binding to address, 283, 287–288  
  client-side sockets, 290  
  closing, 284, 287  
  connecting to end point, 283, 290  
  creating, 284–286  
  definition of, 282–283  
  example of, 289  
  listening, 283, 288  
  permission to use, 339  
  receiving data, 284, 287  
  sending data, 283, 286–287  
  server-side sockets, 287–289  
Sort method  
  arrays, 225  
  lists, 237  
SortedDictionary<TKey, TValue> class, 241  
sorting arrays, 225  
sorting lists, 237  
SourceSwitch class, 454–455  
specialname keyword, IL, 43, 45

- specific cultures, 303, 310**
- spin-locks, 379–380**
- spin-loops, 391–392**
- Split method**
  - Regex class, 487, 488
  - strings, 191–192
- splitting strings, 191–192**
- SSL (Secure Socket Layer), 293**
- SSN (Social Security Number), matching, 465–466**
- STA (Single Threaded Apartment), 390**
- stack**
  - loading values on
    - arguments and locals, 89–90
    - arrays, 99
    - constants, 88–89
    - definition of, 87–88
    - fields, 90
    - indirect loads, 90
  - memory management of, 115
  - storing values from
    - arguments and locals, 89–90
    - arrays, 99
    - definition of, 87–88
    - fields, 90
    - indirect stores, 90
  - for threads, controlling size of, 360–361
- stack crawl, 343**
- Stack Overflow (SO), 118–119**
- stack transition diagram, 84**
- stack-based machines, 84–86**
- StackOverflowException exception, 209**
- stacks, LIFO, 243–244**
- Stack<T> class, 243–244**
- standard devices**
  - communicating through serial port, 282
  - definition of, 280
  - reading from, 281
  - redirecting, 280
  - writing to, 280–281
- standard dictionaries, 239–240**
- standard error (stderr), 280–281**
- standard input (stdin), 280, 281**
- standard lists, 236–239**
- Standard ML language**
  - tail calls and, 94
  - typing strategy of, 13
- standard output (stdout), 280–281**
- starg instruction, IL, 89, 560, 573**
- starvation, 382**
- static assembly loading, 155–156**
- static calls, 93–94**
- static constructors (type constructors), 42**
- static fields**
  - scalar types used for, 19
  - thread static fields, 367–368
- static literal keywords, IL, 28**
- static members, 26**
- static methods, 31**
- static type annotations, 15**
- static typing, 13–16**
- stderr (standard error), 280–281**
- stdin (standard input), 280, 281**
- stdout (standard output), 280–281**
- stelem instruction, IL, 99, 567–568**
- stfld instruction, IL, 90, 568**
- stind instruction, IL, 90, 560–561**
- stloc instruction, IL, 89, 561, 573–574**
- stobj instruction, IL, 561**
- Stopped state, Thread object, 358**
- StopRequested state, Thread object, 358**
- store release, instruction reordering, 388**
- streams**
  - asynchronous I/O using, 261–264
  - buffered streams, 278
  - buffering reads and writes, 260–261
  - closing and disposing stream, 260
  - compressed streams, 278–279
  - definition of, 255–256
  - memory streams, 279–280
  - readability of, 257
  - readers for
    - binary readers, 264, 268–271
    - definition of, 264–265
    - text readers, 264–268
  - reading data using, 256–258, 260–264
  - seeking data using, 259–260
  - writers for
    - binary writers, 264, 268–271
    - definition of, 264–265
    - text writers, 264–268
  - writing data using, 258–259, 260–262
- string freezing, NGen, 165**
- string interning, 88**
- string type, IL, 172**
- StringBuilder class, 202**
- String.Format method, 186**
- StringReader class, 267–268**
- strings**
  - accessing contents of, 186–187
  - books about, 213
  - building, 202
  - case conversions of, 188, 323–325

### strings (continued)

- comparisons, 187–188, 322–325
- concatenating, 185
- definition of, 184–185
- formatting, 186, 196–200
- manipulation of, cultures and, 321–325
- merging, 192
- modifying, 188–192
- padding, 190–191
- parsing, 200–201
- readers for, 267–268
- replacing or removing parts of, 189
- searching, 192
- sorting, 323–325
- splitting, 191–192
- substrings of, 191
- trimming, 189–190
- Unicode used for, 302

**StringWriter class, 267–268**

**strong name of assembly, 133, 135, 141–143, 339**

**strongly typed resources, 318–319**

**StrongNameIdentityPermission class, 339**

**struct constraint, 77**

**struct keyword, C#, 20**

**Structured Exception Handling (SEH), 101**

**structures (structs). See value types**

**stsfld instruction, IL, 90**

**sub instruction, IL, 91, 561–562**

**subclassing**

- inheritance used by, 50–51
- methods and, 35–38
- of types, 49–50

**Substring method, strings, 191**

**substrings, 191**

**subtracted character classes, in regular expressions, 473**

**Sun Java platform, 4**

**SuppressFinalize method, GC class, 203**

**Suspended state, Thread object, 358**

**SuspendRequested state, Thread object, 358**

**Switch class, 454**

**switch instruction, IL, 562**

**sync-block, 21**

**synchronized methods, 371–372**

**SyncLock keyword, VB, 36**

**System ACLs (SACLs), 349**

**system exceptions, 209–210**

**System.Activator class, 505–506**

**System.Array type, BCL, 220–225**

**System.Attribute type, BCL, 64, 515**

**System.Boolean type, BCL, 17, 172, 184**

**System.Byte type, BCL, 17, 172, 181–182**

**System.Char type, BCL, 17, 172, 181, 182**

**System.Collections.ArrayList type, BCL, 72, 247**

**System.Collections.BitArray class, 247**

**System.Collections.CollectionBase class, 247**

**System.Collections.Generic namespace, 225, 226**

**System.Collections.Generic.List<T> type, BCL, 74, 236–239**

**System.Collections.Hashtable class, 247**

**System.Collections.ObjectModel.Collection<T> class, 241–242**

**System.Collections.Queue class, 247**

**System.Collections.SortedList class, 247**

**System.Collections.Stack class, 247**

**System.DateTime type, BCL, 172, 192–195**

**System.Decimal type, BCL, 172, 183–184**

**System.Delegate type, BCL, 60**

**System.Diagnostics namespace, 441**

**System.Double type, BCL, 18, 172, 183**

**System.Enum type, BCL, 65, 69**

**System.Exception type, BCL, 102, 110–111**

**System.FlagsAttribute attribute, 67**

**System.GC class, 202–204**

**System.Globalization.NumberFormatInfo class, 196–200**

**System.Int16 type, BCL, 17, 172, 181**

**System.Int32 type, BCL, 17, 172, 181**

**System.Int64 type, BCL, 17, 172, 181**

**System.IntPtr type, BCL, 18, 172, 192, 405–406**

**System.IO.BinaryReader class, 268**

**System.IO.BinaryWriter class, 268**

**System.IO.Compression namespace, 278**

**System.IO.Compression.DeflateStream class, 278–279**

**System.IO.Compression.GZipStream class, 278–279**

**System.IO.IsolatedStorage namespace, 337**

**System.IO.Ports namespace, 282**

**System.IO.Ports.SerialPort class, 282**

**System.IO.Stream class**

- asynchronous I/O using, 261–264
- buffering reads and writes, 260–261
- closing and disposing stream, 260
- readability of, 257
- reading data using, 256–258, 260–264
- seeking data using, 259–260
- writing data using, 258–259, 260–262

**System.IO.TextReader class, 265**

**System.IO.TextWriter class, 265**

**System.Math class, 205–207**

**System.Net namespace, 282**

**System.Net.Dns class, 290**

**System.Net.DnsPermission class, 338**

**System.Net.NetworkInformation** namespace, 290  
**System.Net.NetworkInformation.Permission** class, 338  
**System.Net.SmtpPermission** class, 338  
**System.Net.SocketPermission** class, 339  
**System.Net.WebPermission** class, 339  
**System.Nullable<T>** type, BCL, 22, 23–24, 93  
**System.Object** type, BCL  
   converting to string, 179  
   definition of, 16–17, 172, 173  
   equality methods for, 173–176  
   finalizers for, 177–179  
   hash codes for, 176  
   type identity of, 179  
**System.Random** class, 207–208  
**System.Reflection.Assembly** type, 157, 498  
**System.Reflection.AssemblyKeyFileAttribute** attribute, C#, 142  
**System.Reflection.AssemblyName** type, 137  
**System.Resources.ResourceManager** class, 319  
**System.Runtime.InteropServices.SafeHandle** class, 412–416  
**System.Runtime.InteropServices.StructLayoutAttribute** attribute, 29  
**System.SByte** type, BCL, 17, 172, 181  
**System.Security.AccessControl** namespace, 348–349  
**System.Security.AllowPartiallyTrustedCallersAttribute (APTCA)**, 332–333  
**System.Security.Cryptography.RandomNumberGenerator** class, 208  
**System.Security.Permissions** namespace, 331  
**System.Security.Permissions.CodeAccessPermission** class, 335  
**System.Security.Policy** namespace, 330, 334  
**System.Security.Policy.CodeGroup** class, 334–335  
**System.Single** type, BCL, 18, 172, 183  
**System.String** type, BCL, 172, 184–185  
**System.Text.Encoding** type, 320–321  
**System.Text.RegularExpressions** namespace, 463, 482  
**System.Text.RegularExpressions.Regex** class, 466, 482–488  
**System.Transactions** namespace, 535  
**System.Transactions.DistributedTransactionPermission** class, 339  
**System.Type** class, 500–502, 505–506  
**System.UInt16** type, BCL, 17, 172, 181  
**System.UInt32** type, BCL, 17, 172, 181  
**System.UInt64** type, BCL, 17, 172, 181  
**System.UIntPtr** type, BCL, 18, 172, 405  
**System.ValueType** type, 16  
**System.Void** type, BCL, 18  
 system-wide culture, 313

## T

\t escape, in regular expressions, 468  
**T&** type, 35  
 tail calls, 94–95  
 tail. prefix, IL, 575  
**TcpClient** class, 292  
**TCP/IP (Transmission Control Protocol/Internet Protocol)**, 291–292  
**TcpListener** class, 292  
**TEB (Thread Environment Block)**, 347, 355, 366  
 temporary files, 276  
 terminate and stay resident programs (TSRs), 353  
 text readers, 264–268  
 text resources, 315–316  
 text writers, 264–268  
 this keyword, C#, 40, 41  
 this pointer, 31  
 thread affinity, 388  
**Thread** class, 356  
**Thread Environment Block (TEB)**, 347, 355, 366  
**Thread Local Storage (TLS)**, 355, 366–367  
**ThreadPool** class, 356–357  
**threads**  
   aborted, exception thrown by, 109–110  
   aborting, 364–365  
   active, obtaining reference to, 356  
   APM (asynchronous programming model), 385–387  
   background threads, 365–366  
   books about, 401–402  
   COM apartments, 390–391, 402  
   concurrency problems with  
     deadlocks, 381–382  
     race condition, 355, 369  
     starvation, 382  
   creating, 359–360  
   critical sections, 387–388  
   definition of, 353–355  
   events and, 382–385  
   explicit management of, 358–366  
   Hyper-Threading (HT) technology, 391–392  
   identity of, 365  
   interrupting, 362–363  
   isolation, levels of, 354–355  
   joining, 363–364  
   managed threads, 355  
   mapped to fibers, 355  
   memory models, 388–390  
   number executing, 355  
   overhead used by, 355  
   pool of, 356–357

## threads (continued)

---

### **threads (continued)**

- priority of, 366, 382
- resuming, 363
- sleeping, 362
- spin-loops, 391–392
- stack size for, controlling, 360–361
- starting, 361
- state of, scheduling, 358–365
- state of, sharing
  - ACLs (Access Control Lists), 374
  - critical sections, 368–369
  - interlocked operations, 378–381
  - lock blocks, 371
  - monitors, 369–372
  - mutexes, 372–373
  - race conditions and, 369
  - reader-writer locks, 374–378
  - semaphores, 373–374
  - synchronized methods, 371–372
- suspending, 363
- thread affinity, 388
- thread start function for, 359–360
- thread static fields, 367–368
- TLS (Thread Local Storage), 355, 366–367
- UI threads, 387

**throw instruction, IL, 101, 568**

**time types, 192–195**

**timers, 385**

**times, regional, 308**

**TimeSpan type, BCL, 195, 208**

**Title method, Console class, 281**

**TLS (Thread Local Storage), 355, 366–367**

**TM (transaction manager), 533–535, 546–548**

**Tokens, assembly metadata, 136**

**tokens, for info APIs, 496, 511–514**

**ToLocalTime method, DateTime type, 195**

**ToLower method, strings, 188, 323–325**

**ToString method**

numbers, 196

Object type, 179, 185, 321–322

**ToUniversalTime method, DateTime type, 195**

**ToUpper method, strings, 188, 323–325**

**Trace class, 444, 449–450**

**TraceFilter class, 444, 456–457, 461**

**TraceListener class, 444, 451–457, 460–461**

**TraceSource class, 443, 445–449, 457–460**

**TraceSwitch class, 456**

**tracing**

architecture of, 441–444

asserts, 446–449, 462

books about, 462

compared to exceptions, 441

configuration of, 457–462

definition of, 440

filtering, 454–457

output of

to console, 452–453

to streams or files, 452–453

Windows Event Log, 452

to XML, 453

**trace listeners**

configuring, 460–461

definition of, 442, 444

filters for, 461

using, 451–457

**trace sources**

conditional compilation for, 450–451

configuring, 457–460

Debug class, 449–450

definition of, 442

Trace class, 449–450

TraceSource class, 445–449

types of, 443–444

**transaction manager (TM), 533–535, 546–548**

**transactions**

ACID properties of, 535

ambient transactions, 536

books about, 548

committed, 534, 537–538

deadlock prevention, 538–539

declarative transactions, 544–546

definition of, 533–535

dependent, 543–544

Dispose method and, 540

distributed, permission to use, 339

example of, 540–541

isolation levels for, 539

managing with Enterprise Services, 544–546

nesting, 541–543

promotion of, 547–548

rolled back, 534, 537–538

scope of

construction of, 537

declaring, 536–537

explicit, 537

two-phase commit (2PC), 548

**translation. See internationalization**

**Transmission Control Protocol/Internet Protocol (TCP/IP), 291–292**

**Trim method, strings, 189–190**

**trimming strings, 189–190**

**trust, CAS, 332–335**

**try/catch blocks**

- catch on Boolean filter, 104–105
- catch on type filter, 102–104
- definition of, 102
- ensuring memory release using, 409
- examples of, 100–101

**TryParse method, strings, 321–322****TSRs (terminate and stay resident programs), 353****two-pass exception model, 101, 111–113****two-phase commit (2PC), 548****type arity, 69–70****type constructors, 42****type forwarding, 160–162****type identity, 97–98, 179****type initializers (type constructors), 42****type instantiation, 70****type members**

- accessibility of, 25–26
- constructors
  - chaining, 40–41
  - default constructors, 39–40
  - definition of, 38–39
  - field initialization in, 41–42
  - type constructors, 42
  - unhandled exceptions in, 42–43
- definition of, 26
- events
  - definition of, 45–46, 382–383
  - delegate chains and, 523
  - exposed by AppDomains, 394
  - monitor-based events, 383–384
  - timers, 385
  - Win32 events, 384–385
- fields
  - constant (literal) fields, 28
  - definition of, 26–27
  - memory layout for structs, controlling, 29–30
  - read-only fields, 27–28
  - size of, 27
- instance members, 26
- methods
  - abstract methods, 51
  - argument passing style of, 33–35
  - arguments of, 30, 89–90
  - definition of, 30–31
  - exception handlers in, 38
  - instance methods, 31
  - loading and storing from stack, 89–90
  - locals of, 31–32
  - new slots, 37–38
  - output parameters, 34

- overloading, 32
- overriding, 36–37
- parameters of, 30
- return parameter of, 30
- static methods, 31
- subclassing and, 35–38
- variable argument methods, 35
- virtual methods, 36–37, 93–94, 96–97, 128

**properties**

- definition of, 43–44
- indexer properties, 44–45
- mixed mode accessibility for, 45
- static members, 26
- visibility of, 25–26

**type safety. See also CTS (Common Type System)**

- collections and, 72–73
- enumerations and, 68–69
- example of unsafe code, 12–13
- importance of, 11
- verification of, 9, 11–12, 86

**type systems. See also CTS (Common Type System)**

- books about, 78–79
- bridging, 434–436
- definition of, 9
- static compared to dynamic, 13–16

**type unification, 16–18****type verification, 9, 11–12, 86****TypeDef section, assembly metadata, 136, 141****TypeLoadException exception, 210****TypeRef section, assembly metadata, 136, 141****types. See also delegates; primitive types; reference types (classes); value types (structures)**

- accessibility of, 25–26, 45
- blittable types, 434
- building, 530
- constructors for
  - chaining, 40–41
  - default constructors, 39–40
  - definition of, 38–39
  - field initialization in, 41–42
  - type constructors, 42
  - unhandled exceptions in, 42–43
- custom attributes (CAs), 64–65, 496, 514–519
- enumerations
  - definition of, 65–67
  - flags-style enumerations, 67–68, 69
  - helper methods for, 69
  - type safety and, 68–69
- info APIs for, 500–502
- nested, 26
- organizing into namespaces, 58–60

## types (continued)

- polymorphism and, 49–50
- runtime type checking, 58
- sealing, 57
- subclassing, 49–50
- visibility of, 25–26

## U

- `\u` escape, in regular expressions, 469
- UDP (User Datagram Protocol), 292–293**
- UdpClient class, 293**
- UI culture, 312–313**
- UI, permission to access, 338**
- UI threads, 387**
- uint16 type, IL, 172**
- uint32 type, IL, 172**
- uint64 type, IL, 172**
- UIPermission class, 338**
- unaligned. prefix, IL, 575**
- unbox instruction, IL, 93, 568**
- unboxing**
  - of collections, 73–74
  - definition of, 23
  - of values, 93
- undeniable exceptions, 109–110**
- underflow, floating points, 183**
- unhandled exceptions**
  - in constructors, 42–43
  - definition of, 99, 108–109
- Unicode character encoding**
  - books about, 493
  - definition of, 302
- Unicode properties, in regular expressions, 472–473**
- unification of objects and values, 22–24**
- Uniform Resource Locator (URL), 293**
- UnionCodeGroup class, 334**
- unmanaged code, 403. See also interoperability; managed code**
- UnManagedMemoryStream class, 280**
- unsigned mutexes, 372**
- unsigned int8 type, IL, 17, 172**
- unsigned int16 type, IL, 17**
- unsigned int32 type, IL, 17**
- unsigned int64 type, IL, 17**
- unsigned native int type, IL, 18, 405**
- Unstarted state, Thread object, 358, 359**
- untrusted code, CAS, 332**
- URL (Uniform Resource Locator), 293**
- UrlIdentityPermission class, 340**
- User Datagram Protocol (UDP), 292–293**

## User policy level, 341

- user-based security**
  - access controls, 348–351
  - impersonation, 348
  - simple authentication, 347–348
  - user identity for, 347–348
- using keyword, C#, 59**
- UTF-7 encoding, 320**
- UTF-8 encoding, 320**

## V

- `\v` escape, in regular expressions, 468
- value types (structures)**
  - allocating and initializing, 92–93
  - compared to reference types, 18
  - construction of, 39–40
  - creating, 20
  - definition of, 16, 18–20
  - memory layout for, controlling, 29–30
  - size of, determining, 22
  - when to use, 18–19
- values**
  - accessing raw value of boxed value (unboxing), 23
  - definition of, 18, 20
  - memory layout for, 21–22
  - transforming into objects (boxing), 23
  - unification with objects, 22–24
- vararg keyword, IL, 35**
- variable argument methods, 35**
- VB (Visual Basic) language**
  - books about, 79
  - conditional compilation for, 451
  - default constructors, 39
  - generics support, 71
  - managed code written in, 5
  - operators, list of, 48–49
  - primitive types, list of, 172
  - support for, 16
  - `SyncLock` keyword, 371
  - typing strategy of, 14, 15
- VBA (Visual Basic Automation), 4**
- vectors, 216–217, 225**
- verification of type safety, 9, 11–12, 86**
- version 2.0 improvements, 7**
- virtual calls, 93–94**
- virtual execution environment, 5. See also CLR (Common Language Runtime)**
- virtual keyword, C#, 36**
- virtual machine, 5. See also CLR (Common Language Runtime)**

**virtual methods**

- calling, 93–94, 128
- definition of, 36–37
- nonvirtual calls to, 96–97

**virtual table (vtable), 125**

**virtualizing architectures, books about, 131–132**

**visibility of types, 25–26**

**Visual Basic Automation (VBA), 4**

**Visual Basic language. See VB language**

**void type, IL, 18**

**volatile. prefix, IL, 575**

**vtable (virtual table), 125**

**W**

**\w class, in regular expressions, 466, 471**

**WaitForPendingFinalizers method, GC class, 203**

**WaitHandle class, 357**

**WaitSleepJoin state, Thread object, 358, 362, 363**

**weak references, 204–205**

**weakly typed collections, 246–247**

**weakly typed resources, 319**

**Web resources, permission to use, 339**

**web services, 4**

**WebClient class, 294–295**

**wildcards, in regular expressions, 464**

**Win32 events, 384–385**

**Windows APIs, history of, 3–4**

**Windows Event Log**

- definition of, 439
- permission to access, 338
- tracing output written to, 452

**Windows Forms, 4, 5**

**Windows types, 434**

**Windows.h file, 136**

**WinFX, 5**

**workstation garbage collection, 122**

**WOW64 (Windows-on-Windows64), 7**

**wrapped exceptions, 108**

**Write method**

- binary data, 270
- NetworkStream class, 286–287
- Stream class, 258–259

**WriteByte method, Stream class, 258**

**WriteLine method, strings, 267**

**writers, 255–256**

**X**

**\x escape, in regular expressions, 469**

**X509Certificate, 339**

**XML (Extensible Markup Language)**

- APIs, 5
  - parsing with regular expressions, 480–481
  - for resource files, 316–317
- xor instruction, IL, 91, 562**

**Z**

**\Z meta-character, in regular expressions, 475**

**zero page list, 117**

**\0 escape, in regular expressions, 468**

**ZoneldentityPermission class, 340**