

Index

• Symbols •

+ (add) operator, 50
| (and/or true) operator, 55–56
'\' (backslash) character, 38
- (binary) subtract operator, 50
& (both are true) operator, 55–56
&& (both are true/short-circuit) operator, 55–56
'r' (carriage return) character, 38
/ (division) operator, 50
. (dot) operator, 155
^ (either is true, not both/exclusive or) operator, 55–56
|| (either is true, not both/short-circuit) operator, 55–56
" " (empty string), 39
== (equals) operator, 120
! (false/not) operator, 55
> (greater than) operator, 53
>= (greater than or equal to) operator, 53
=> (lambda) operator, 356
< (less than) operator, 53
<= (less than or equal to) operator, 53
% (modulo) [modulo symbol] operator, 50
* (multiply) operator, 50
'n' (new line) character, 38
!= (not equal to) operator, 53, 120
'0' (null) character, 38
== (same value) operator, 53
{0:# or 0 [modulo]} specifier, `Format()` method, 141
{0:#...} specifier, `Format()` method, 140
{0:0...} specifier, `Format()` method, 140
't' (tab) character, 38
:? (ternary) operator, 67
- (unary) (take negative of) operator, 50

• A •

abstract classes
 definition, 297–300
 interfaces, 305, 324–329
 UML, 293
abstract keyword, 305
abstract methods, 305
`AbstractBaseClass` class, 299–300
`AbstractInheritance` program, 298–299
abstraction concept in object-oriented programming, 209–211
access control
 class members, 219–228
 classes' responsibility for, 213–214, 215
 current object access, 198–204
 delegates, 340
 errors in code, 409–410
 interface tool for, 319–321
 keywords, 219–224, 340
 methods, 220–224
 object members, 152, 195
 in object-oriented programming, 214, 215
accessors (access methods), 228, 230, 231
actions on arrays, 358
`Action<T>` delegate, 359–360
`Add()` method, 106, 108
adding numbers array, 398
`AddRange()` method, 107
ADO.NET, 371
`Aggregate()` method, 399–400
aggregation queries, 397–400
`AlignOutput` program, 134–136
anonymous methods, 345–346, 365
anonymous objects, 368
anonymous types, 47, 380–382
`Append()` method, 145
Application Wizard, 15–19

- arguments, method
 - default, 175–177
 - inheritance, 268
 - initializing variables, 182
 - introduction, 169–170
 - matching definitions with usage, 173–174
 - overloading, 174–177
 - passing multiple, 172–173
 - passing single, 170–171
 - reference-type, 178–181, 191–193
 - returning values, 182–186
 - value-type, 177–180
- arithmetic operators, 49–53
- Array class
 - compared to collection classes, 106
 - initializing, 110
 - introduction, 90–96
 - searching for values with, 123
 - sorting elements, 97–101
 - var keyword with, 102
- arrays
 - actions on, 358
 - adding numbers type, 398
 - data type limitations, 103
 - Find() method, 355–359
 - foreach loop, 96–97
 - ForEach() method, 358–361
 - for loop, 91–92, 96–97
 - initializing, 96, 102, 110
 - interface in, 308, 309, 311–313, 317
 - introduction, 90–96
 - joining array string, 133–134
 - methods, 103–104, 107
 - null items, 106
 - properties in, 103–104
 - querying, 371
 - reference-type variables, 157
- as operator, 264–265
- AsReadOnly() method, 321
- assembly, definition, 223
- assigning values to variables, definition, 29
- assignment operator
 - definition, 28
 - error location, 59
 - function and evaluation of, 51–52
 - reference types, 155
- Average() method, 183–184
- AverageAndDisplay() method, 172–175, 185–186

• B •

- BankAccount class
 - access control, 220–224
 - inheritance, 255–257, 258–260, 262–263, 271–273
 - initializing objects, 240
- base class
 - C# prohibition on multiple inheritance, 307
 - overloading (hiding), 277–283
 - subclass relationship, 275
- base interface, 322
- base keyword, 269–270, 282, 283
- BaseClass class, 253, 266–270, 284
- BigInteger variable type, 31
- binary arithmetic, 56
- binary compared to unary operators, 50
- bitwise operators, 56
- block statements, lambda, 363–364
- bool value-type variable
 - constants for, 44
 - conversion restriction, 45, 59
 - in if statement, 62
 - introduction, 37
 - in loop expression, 73
 - operators, 53–56, 264
- boxing process, 116, 185
- braces in expressions, 62, 73, 415
- break command
 - ending loops, 77–78
 - in lambda expressions, 364
 - in nested loops, 85
 - switch statement, 71–72
- breakpoint, setting, 237
- bubble sort, 98–101
- Build-Build, 18
- BuildASentence program, 122–125
- byte integer type, 30

• C •

- C-currency specifier, Format() method, 140
- C# programming language
 - advantages of, 12–13
 - application creation basics, 15–19
 - book overview, 3
 - definition, 12

- implementation of object-oriented programming, 215
- inferring data types, 46–47
- as Internet tool, 13–14
- introduction, 1–8, 11
- multiple inheritance prohibition, 307
- new features, 2
- notational, 121
- query features, 368
- reviewing console application, 20–22
- running console application, 19–20
- saved text in Toolbox, 22–24
- security advantages, 13
- Visual C#, 14–15
- Web site resources, 8
- CalculateInterest program, 63–64
- CalculateInterestTable programs, 73, 78–82, 163–169
- CalculateInterestWithEmbeddedTest, 67–69
- call-by-reference feature, 182–186
- callback method, 333–345, 346
- camel-casing naming style, 42
- CAN_BE_USED_AS relationship, 303–304, 310–317
- capitalization, importance in code, 19
- case, text
 - capitalization importance in code, 19
 - ignoring with `Compare()`, 124
 - in naming conventions, 42
 - Pascal-cased naming style, 42
 - switching, 86–87, 125–127
- casts, 45, 58, 262–265
- char variable type
 - compared to string, 40–41
 - constants for, 44
 - as intrinsic variable, 118
 - introduction, 38
- character variable types, 37–40, 44, 71, 118.
 - See also* string variable type
- class-from-class inheritance, 252–253
- classes
 - See also* collection classes
 - See also* constructors
 - See also* inheritance
 - See also* properties
 - abstract, 293, 297–300, 305, 324–329
 - AbstractBaseClass, 299–300
 - access control, 213–214, 215, 219–228
 - Array, 90–96, 97–101, 102, 106, 110, 123
 - BankAccount, 220–224, 240, 255–257, 258–260, 262–263, 271–273
 - base-level, 275, 277–283, 307
 - BaseClass, 253, 266–270, 284
 - Console, 189
 - Convert, 129–131
 - DataSet, 321
 - definition, 150–151, 154
 - dependency, 324
 - Dictionary<TKey, TValue>, 104, 105, 108–110, 111
 - discrimination between objects, 154–155
 - extension methods, 205–208
 - factoring, 292–297
 - grouping different data types, 149
 - HashSet<T>, 2, 104, 112–115
 - initializing, 110–111, 112, 240
 - instances, compared to, 212
 - interfaces, 305, 317–319
 - intrinsic data type, compared to, 149, 152
 - List<T>, 89, 104, 105–108, 110, 111, 114, 354, 355, 357–359
 - member definition, 151, 159, 162
 - methods, relationship to, 161, 162, 191
 - naming conventions, 151, 197
 - Object, 116, 151–152, 153–154, 265–266, 291
 - objects, compared to, 232
 - in overloaded methods, 277
 - property definition, 159, 228–232
 - querying, 371
 - Queue<T> class, 104, 110
 - referencing to other classes, 157–158
 - SavingsAccount, 258–260, 262–263, 273
 - sealing, 300–301
 - Stack<T>, 104, 110
 - static methods, 202–204, 343, 357, 392
 - String, 117, 120, 127–129, 139–143, 205–208
 - string variable type as, 117
 - Subclass, 253, 266–270, 284
 - subclasses, 299–300, 325
 - substitutable, 262
 - System.EventArgs, 350
 - System.MulticastDelegate, 336
 - TimeSpan, 42
 - UML syntax, 293

- classification concept in object-oriented programming, 211–213
- `ClassMethod()` method, 162
- client, terminology issues, 327
- CLR (.NET Common Language Runtime), 3
- code region, introduction, 17
- Code Snippets, 22
- collection classes
 - data type limitations, 103
 - `Dictionary`, 108–110
 - initializing, 110–111, 112, 368
 - interface in, 308, 309
 - `List`, 105–108
 - methods in, 103–104
 - `Object`, 116
 - properties in, 103–104
 - querying, 371, 372
 - returning reference types to, 321
 - syntax, 104–105
- collection initializers, function of, 111, 368
- collections. *See also* arrays; collection classes
 - converting query results to, 374
 - introduction, 89
 - reference type variables, 157
 - sets, 112–115
- command-line input, `string` variable type, 128–134
- Command Prompt window, executing programs from, 20
- comments, 21, 163
- `Compare()` method, 121–125, 126
- `CompareTo()` method, 312–313, 316, 317
- comparing numbers, floating-point variable limitations, 35
- compiler, C#, inferring data types, 46–47
- compiling, definition, 16
- compound logical operators, 55–56
- computers, basic concepts, 11–12
- `Concat()` method, 136–138
- conceptual compared to procedural view of OO programming, 210–213
- condition statement, 83
- conditional expression (`if` statement)
 - `bool` variable type, 62
 - `else` statement, 65–67
 - embedded, 67–69
 - `goto` statement, 86–87
 - introduction, 62–65
 - nesting of, 67–69
 - `switch` construct, 69–72
- console application
 - initial creation of, 15–19
 - reviewing, 20–22
 - running basic, 19–20
- `Console` class, 189
- `const` data member, 160
- constants
 - `const` data member, 160
 - importance of defining, 69
 - `string` immutability, 119
 - as variable types, 31
- constructor chaining, 249
- constructors
 - duplication problem solving, 246–249
 - inheritance, 266–273
 - initializers, 239–243
 - interfaces, 215, 322
 - introduction, 232–233
 - local object creation, 249–250
 - overloading, 243–246
 - property, 228–232
 - replacing default, 234–239
 - `switch`, 69–72
- `ConstructorSavingsAccount` program, 271–273
- containment, 259–260, 324, 325–329
- `Contains()` method, 107, 128
- `ContainsKey()` method, 109
- `ContainsValue()` method, 109
- `continue` command, loops, 77–78
- conversion, type (casts), 45, 58, 262–265
- `Convert` class, 129–131
- `ConvertTemperatureWith Float` program, 34
- `ConvertTemperatureWith RoundOff` directory, 32
- `Convert.ToDecimal()` command, 64
- `Count` property, 107, 109
- counting number type, 34, 38, 45
- counting query results, 398
- current object, 196, 198–204

• **D** •

- D–decimal specifier, `Format()` method, 140
- data members
 - accessing with objects, 195
 - compared to properties, 229
 - `const`, 160
 - initializing, 240
 - interface restriction against, 305
 - naming conventions, 202
 - nonstatic, 240
 - `readonly`, 160
- data types. *See also* arrays; classes; variables
 - anonymous, 47, 380–382
 - array and collection class limitations, 103
 - assumptions, 43
 - conversion rules, 45, 58, 262–265
 - declared compared to real, 285
 - delegates as, 336
 - errors in type assignment, 59
 - importance of, 152
 - inferring, 46–47
 - initializing collections, 111
 - introduction, 27
- database queries, 369–370, 371
- `DataSet` class, 321
- `DataGridView` object, 321
- `Debug-Start`, 18
- `Debug-Start Without Debugging`, 19
- debugger, executing constructor from, 237
- decimal variable type, 35–37, 45
- `DecimalBankAccount` program, 226–228
- declared compared to real (run-time) data type, 285
- default constructor, 233
- default program location, 17
- deferred execution, queries, 370, 375
- `delegate` keyword, 346
- delegates. *See also* lambda expressions
 - anonymous methods, 345–346
 - callback problem, 334–345
 - events, 346–352
 - in query expressions, 392, 396–397
- `DemonstrateCustomConstructor` program, 235–237, 241
- demotion, variable, 58
- dependency, class, 324
- Design Patterns For Dummies* (Holzner), 326, 347
- destructor method, 274
- `Dictionary<TKey, TValue>` class
 - function of, 104
 - initializing, 111
 - notation aspects, 105
 - using, 108–110
- digits of accuracy for floating-point variable, 33–34
- `Display()` method, 311–312
- `DisplayRatio()` method, 186
- distributed development, 13
- DLINQ, 371
- do-to-each problem, 353–355
- DOS command line, executing programs from, 20
- double slashes, comments, 21
- double value-type variable
 - constants for, 44
 - decimal variable, compared to, 36
 - `out` keyword with, 185
 - Pentium processor, 35
 - size and range, 33–34
- `DoubleBankAccount` program, 225–226
- `doublesArray` variable, 91–92
- downcast, 58
- duplication of constructors, 246–249
- Dykes, Lucinda (author), *XML For Dummies*, 370

• **E** •

- E–exponential specifier, `Format()` method, 140
- early binding, 285
- `else` statement, 65–67, 69
- embedded `if` statement, 67–69
- empty string, 39, 128, 187–188
- errors, code
 - access control, 409–410
 - executables without termination, 411–412

errors, code (*continued*)

- inheritance of classes, 412–413
- initializing of variables, 410–411
- interface problems, 413–414
- introduction, 405
- method problems, 414–415
- missing braces, 415
- type assignment, 59
- variable names, 406–407
- variable type conversion, 407–409
- event keyword, 347
- EventHandler delegate type, 348, 350–352
- events, 346–352
- ExceptWith() method, 114
- .exe file extension, 12
- executable program, 12
- execution loop program. *See also* foreach loop
 - breaking, 77–78, 85
 - “do-to-each” trick, 291
 - do...while, 76–77, 78–82
 - for, 82–84, 91–92, 96–97
 - infinite, 76, 84
 - introduction, 72–76
 - logical variable type, 73, 84
 - nesting, 85–86
 - scope rules, 82
 - string manipulation, 137–138
- explicit compared to implicit use of this keyword, 200
- explicit demotion, 58
- explicit type conversion (cast), 45, 58, 262–265
- expression compared to statement lambda types, 363–364
- extended name in overloaded methods, 276–277
- extension methods. *See also* inheritance function of, 368
 - introduction, 205–208
 - in query expressions, 373, 376, 391–397
- extensions, file, 12
- external compared to internal methods, 224

• F •

- F-fixed specifier, Format() method, 140
- factorials, 31, 399–400
- factoring of classes, 292–297
- false value for bool variable, 37
- filter expression in queries, 373, 377–379
- Find() method, 355–358
- fixed-length (value-type) variables. *See also* bool value-type variable; floating-point variable type; int (integer) value-type variable
 - basic method, 27–28
 - char, 38, 40–41, 44, 118
 - character, 37–40, 44, 71, 118
 - as compared to is operators with, 265
 - conversion of, 44–45
 - DateTime type, 41–43
 - decimal, 35–37, 45
 - definition, 40
 - double, 33–34, 35, 36, 44, 185
 - fractions, 31–37
 - inferring data types, 46–47
 - introduction, 27
 - in method arguments, 177–180
 - numeric constants, 43–44
 - real numbers, 32–35
- fixed-value array, 90–92, 94
- FixedArrayAverage program, 90–92, 94
- float value-type variable, 33, 44
- floating-point variable type
 - conversion to counting, 45
 - decimal, compared to, 35–37
 - double, 33–34, 35, 36, 44, 185
 - introduction, 32–35
 - limitations, 34–35
 - logical comparison, 54–55
 - modulo operator restriction, 50
 - switch statement, 71
 - types, 33
- Fold() method, 399
- fonts, char variable limitation, 38
- for(;;) infinite loop, 84
- for loop, 82–84, 91–92, 96–97
 - logical operators in, 73

- foreach loop
 - Dictionary<TKey, TValue> class, 109
 - do-to-each problem, 353
 - interfaces, 310
 - introduction, 96–97
 - queries, 370, 373, 375, 383–385, 386
 - with strings, 126–127
- ForEach() method, 354, 355, 358–361
- Format() method, 139–143
- format string, definition, 139
- FORTRAN, 170
- fractions, declaring variables, 31–37
- from keyword, 373, 392
- Func<T, U> delegate type, 392, 396
- function pointers (delegates). *See also* lambda expressions
 - anonymous methods, 345–346
 - callback problem, 334–345
 - events, 346–352
 - in query expressions, 392, 396–397
- functions (methods). *See also* arguments, method; properties; static (class) methods
 - abstract, 305
 - access control, 220–224
 - accessors, 228, 230, 231
 - Add(), 106, 108
 - AddRange(), 107
 - Aggregate(), 399–400
 - anonymous, 345–346, 365
 - Append(), 145
 - in arrays and collection classes, 103–104, 107
 - AsReadOnly(), 321
 - Average(), 183–184
 - AverageAndDisplay(), 172–175, 185–186
 - callback, 333–345, 346
 - classes, relationship to, 161, 162, 191
 - ClassMethod(), 162
 - Compare(), 121–125, 126
 - CompareTo(), 312–313, 316, 317
 - Concat(), 136–138
 - Contains() method, 107, 128
 - ContainsKey(), 109
 - ContainsValue(), 109
 - customizing with delegates, 337–345
 - definition, 161–163
 - delegates, 334–345
 - destructors, 274
 - Display(), 311–312
 - DisplayRatio(), 186
 - errors in code, 414–415
 - example for reference, 163–169
 - ExceptWith(), 114
 - extension, 205–208, 368, 373, 376, 391–397
 - external compared to internal, 224
 - Find(), 355–358
 - Fold(), 399
 - ForEach(), 354, 355, 358–361
 - Format(), 139–143
 - generic, 395–396
 - GroupBy<T>, 392
 - hiding, 277–283, 287–288, 319–321
 - history of, 170
 - IIntersectWith(), 115
 - implemented (nonabstract), 305
 - IndexOf(), 107, 127
 - IndexOfAny(), 127, 138
 - Init(), 247, 249, 257
 - Insert(), 145
 - InstanceMethod(), 162
 - Int32.TryParse(s, n), 131
 - interfaces, 305, 308–309, 319–321
 - IntersectWith(), 114
 - introduction, 161–163
 - InvokeMethod(), 195–196
 - IsAllDigits(), 130–131, 133
 - IsNullOrEmpty(), 128
 - Join(), 133–134
 - LastIndexOf(), 127
 - Left(), 206–207
 - List<T> class, 107–108
 - Main(), 21, 135–136, 235, 257, 258, 276, 316–317
 - naming conventions, 163, 175, 197
 - nested loops, 86
 - nonstatic (instance), 191–208, 357
 - nonvoid compared to void, 186
 - OrderBy<T>, 392, 393
 - overloading inherited, 276–283
 - Pad(), 134–136

functions (methods) *(continued)*

PadRight(), 135–136
 in query expressions, 373, 376, 391–397
 Read(), 154
 ReadLine(), 64, 123, 154
 Remove(), 145
 Replace(), 136–138
 Round(), 75, 226–228
 security, 220–224
 Select<T>, 392, 398
 Slice(), 207–208
 Sort(), 100, 317
 Split(), 131–133, 138–139
 static compared to nonstatic, 193–197
 String class, 127–129, 205–208
 Substring(), 126, 206–208
 SymmetricExceptWith(), 115
 ToArray(), 107
 ToBoolean(), 129
 ToDouble(), 129
 ToFloat(), 129
 ToInt32(), 120
 ToInt64(), 129
 ToLower(), 125–126
 ToString(), 126, 145, 291
 ToUpper(), 119, 125–126
 Trim(), 128–129, 134–136
 UnionWith(), 113, 115
 Update(), 180–181, 184
 value of, 170
 virtual, 288–290
 Where<T>, 392, 393, 396–397
 WriteLine(), 64, 109, 140, 189, 311

• G •

garbage collector, 156–157, 274
 generators, query, 373, 392
 generic collections, 105–106, 116
 generic methods, 395–396
 get section, 229, 230, 231
 goto statement, 86–87, 364
 group by operator, 373, 374, 383–389, 392
 GroupBy<T> method, 392
 grouping query results, 373, 374,
 383–389, 392
 GUI (Graphical User Interface), 14

• H •

HandleTwo delegate, 361–362
 HAS_A relationship, 324, 325–329
 hash table, definition, 108
 HashSet<T> class, 2, 104, 112–115
 hiding methods, 277–283, 287–288, 319–321
 HidingWithdrawalPolymorphically
 program, 284–285
 high-level computer languages, 12
 Holzner, Steve (author), *Design Patterns For
 Dummies*, 326, 347
 Hungarian notation, 42

• I •

IComparable<T> interface, 310, 312–313,
 316
 IDisplayable interface, 311–312,
 316–317
 IEnumerable<T> interface, 310, 378, 392,
 395
 if statement
 bool variable type, 62
 else statement, 65–67
 embedded, 67–69
 goto statement, 86–87
 introduction, 62–65
 nesting of, 67–69
 switch construct, 69–72
 IIntersectWith() method, 115
 immutability of strings, 119
 implementation of interfaces, 305–306
 implemented methods (nonabstract), 305
 implicit compared to explicit use of this
 keyword, 200
 implicit promotion, operators, 58
 implicit type conversion, 45, 59
 increment operator, 52–53, 84
 indentation in code, value of, 65
 index, array, 91
 IndexOf() method, 107, 127
 IndexOfAny() method, 127, 138
 IndexOutOfRangeException error
 message, 94

- indirection
 - abstract classes, 293, 297–300, 305, 324–329
 - definition, 327–329
- inferring data types, 363
- infinite loop, 76, 84
- inheritance
 - abstract classes, 297–300
 - class-from-class, 252–253
 - conceptual relationship, 291–292
 - constructors, 266–273
 - destructor, 274
 - errors, code, 412–413
 - factoring of classes, 292–297
 - functions of, 254–257, 275
 - HAS_A relationship, 324, 325–329
 - interfaces, 303–304, 305, 307, 322
 - introduction, 251
 - invalid casts at run time, 262–265
 - IS_A relationship, 253, 257–261, 262, 304
 - maintaining correct relationships, 294–296
 - Object class, 265–266
 - overloading methods, 276–283
 - sealing classes, 300–301
 - substitutable classes, 262
 - ToString() method, 291
- InheritanceExample program, 252
- InheritingAConstructor program, 266–267
- Init() method, 247, 249, 257
- initializers, 239–243, 368
- initializing
 - arrays, 96, 102, 110
 - automatic calls through constructors, 232–233, 236
 - classes, 110–111, 112, 240
 - collection classes, 110–111, 112, 368
 - definition of variable, 29
 - errors in variable, 410–411
 - importance for variable usage, 182
 - object-initializer syntax, 242–243
- Insert() method, 145
- InstanceMethod() method, 162
- instances. *See also* objects
 - classes, compared to, 212
 - definition, 152
 - delegate, 336
 - members, 159
 - methods, nonstatic, 162–163, 191–208, 357
 - static methods, compared to, 162
- instantiating, definition, 152
- int (integer) value-type variable
 - BigInteger, 31
 - byte, 30
 - compared to decimal, 35–37
 - constants for, 44
 - conversion of, 45, 58
 - definition, 28–29
 - fraction problems of, 32
 - long, 30, 44, 45
 - Object class problems, 116
 - parsing numeric input, 129
 - sbyte, 30
 - short, 30
 - signed compared to unsigned, 31, 44
 - types of, 30
 - uint, 30
 - Ulong, 30
 - ushort, 30
- int keyword, 102
- Int32.TryParse(s, n) method, 131
- interface keyword, 215, 305
- interfaces
 - CAN_BE_USED_AS, 303–304, 310–317
 - constructors, 215, 322
 - definition, 305
 - errors in code, 413–414
 - functions of, 308–310
 - HAS_A, 325–329
 - hiding methods from, 319–321
 - implementation, 305–306
 - inheritance, 303–304, 305, 307, 322
 - introduction, 303
 - management of OO program changes, 323–325
 - naming conventions, 306
 - predefined, 310
 - purpose of, 306
 - terminology overlap, 308
 - unifying class hierarchies, 317–319
 - usable in object-oriented programming, 213–214
 - usefulness of, 307–308

internal compared to external methods, 224

internal keyword, 223, 250

Internet, C# as programming tool for, 13–14

intersection operation, sets, 112

IntersectWith() method, 114

into operator, 373, 386, 387–389

intrinsic variables. *See also* bool value-type variable; int (integer) value-type variable

char, 38, 40–41, 44, 118

classes, compared to, 149, 152

definition, 40

double, 33–34, 35, 36, 44, 185

InvokeBaseConstructor program, 269–270

InvokeMethod() method, 195–196

is operator, 263–264, 287–288

IS_A relationship, 253, 257–261, 262, 304

IsAllDigits() method, 130–131, 133

IsNullOrEmpty () method, 128

iterable sequence for queries, 373

iterator blocks, 397

• J •

Java programming language, 2, 13–14

Join() method, 133–134

join operator, 373

“jump” (return) statement

- constructor restriction, 235
- extended name, 276
- interface as, 308
- lambda expression, 364
- method argument values, 182–186
- nested loops, 86
- query expressions, 375
- reference types to collection classes, 321

• K •

Keys property, 109

KeyValuePair<TKey, TValue>, 109

keywords

- abstract, 305
- access control, 219–224, 340
- base, 269–270, 282, 283
- delegate, 346
- event, 347

- from, 373, 392
- int, 102
- interface, 215, 305
- internal, 223, 250
- out, 179–180, 184–185
- override, 288–290, 305
- private, 220–222, 293
- protected, 223, 273, 351
- public, 219–224, 293, 305
- in query expressions, 372–373, 377–391
- ref, 179–180, 184
- sealed, 300–301
- security, 219–224, 340
- this, 199–204, 248, 282–283
- var, 46–47, 102, 363, 368, 372, 382
- virtual, 288–290, 305, 351
- void, 185–186, 364

• L •

label, goto statement, 86

lambda expressions

- definition, 352
- delegates, 365
- do-to-each problem, 353–355
- Find() method examples, 355–362
- function of, 368
- in queries, 393
- syntax details, 362–364

Language Integrated Query (LINQ), 2, 367.

- See also* lambda expressions; query expressions
- adding numbers array, 398
- C# features using, 368
- counting product, 398
- definition and functions, 368–372
- factorial computing, 399–400
- filter expression, 377–379
- inferring data types, 47
- introduction, 2, 367
- methods, 391–397
- overview example, 372–376
- reflection, 401
- selecting and grouping results, 379–389
- sorting results, 389–391
- summing, 399
- uses for, 376–377

LastIndexOf () method, 127

LastIndexOfAny () method, 127

- late binding (polymorphism)
 - declared type, decision to use, 285–287
 - “do-to-each” loop trick, 291
 - interfaces, 322
 - introduction, 284–285
 - is operator and hidden methods, 287–288
 - overview, 215
 - virtual method and overriding, 288–290
 - Left() method, 206–207
 - Length property, array, 95–96
 - let operator, 373
 - libraries, Java, 14
 - LINQ (Language Integrated Query). *See also* lambda expressions
 - adding numbers array, 398
 - C# features using, 368
 - counting product, 398
 - definition and functions, 368–372
 - factorial computing, 399–400
 - filter expression, 377–379
 - inferring data types, 47
 - introduction, 2, 367
 - methods, 391–397
 - overview example, 372–376
 - reflection, 401
 - selecting and grouping results, 379–389
 - sorting results, 389–391
 - summing, 399
 - uses for, 376–377
 - LINQ to Objects feature, 371
 - list of items, 89, 157, 374. *See also* arrays; collection classes
 - listeners for events, 347
 - List<int> variable, 360
 - List<T> class
 - definition, 89
 - HashSet<T> class, 114
 - initializing, 110, 111
 - introduction, 105–108
 - lambda expression, 354, 355, 357–359
 - overview, 104
 - literal, string, 30
 - local variables, 232–233, 364
 - logical variable type (bool)
 - constants for, 44
 - conversion restriction, 45, 59
 - in if statement, 62
 - introduction, 37
 - in loop expression, 73, 84
 - operators, 53–56, 84, 264
 - long int integer type, 30, 44, 45
 - loops. *See also* foreach loop
 - breaking, 77–78, 85
 - “do-to-each” trick, 291
 - do...while, 76–77, 78–82
 - for, 82–84, 91–92, 96–97
 - infinite, 76, 84
 - introduction, 72–76
 - logical variable type, 73, 84
 - nesting, 85–86
 - scope rules, 82
 - string manipulation, 137–138
 - lowercase strings, 125–127
- M •
- machine language, 11
 - Main() method
 - constructors with, 235
 - formatting output, 135–136
 - interface implementation, 316–317
 - introduction, 21
 - overloaded method function, 276
 - reduced role with inheritance, 257, 258
 - Mansfield, Richard (author), *Visual Basic .NET Database Programming For Dummies*, 370
 - maritalStatus program, 70–72
 - mathematical set operations on query results, 374
 - members. *See also* data members
 - class, 151, 159, 162, 219–228
 - instances, 159
 - object, 152, 195
 - method calls, definition, 64
 - methods. *See also* arguments, method; properties; static (class) methods
 - abstract, 305
 - access control, 220–224
 - accessors, 228, 230, 231
 - Add(), 106, 108
 - AddRange(), 107
 - Aggregate(), 399–400
 - anonymous, 345–346, 365
 - Append(), 145
 - in arrays and collection classes, 103–104, 107

methods (*continued*)

- AsReadOnly(), 321
- Average(), 183–184
- AverageAndDisplay(), 172–175, 185–186
- callback, 333–345, 346
- classes, relationship to, 161, 162, 191
- ClassMethod(), 162
- Compare(), 121–125, 126
- CompareTo(), 312–313, 316, 317
- Concat(), 136–138
- Contains() method, 107, 128
- ContainsKey(), 109
- ContainsValue(), 109
- customizing with delegates, 337–345
- definition, 161–163
- delegates, 334–345
- destructors, 274
- Display(), 311–312
- DisplayRatio(), 186
- errors in code, 414–415
- example for reference, 163–169
- ExceptWith(), 114
- extension, 205–208, 368, 373, 376, 391–397
- external compared to internal, 224
- Find(), 355–358
- Fold(), 399
- ForEach(), 354, 355, 358–361
- Format(), 139–143
- generic, 395–396
- GroupBy<T>, 392
- hiding, 277–283, 287–288, 319–321
- IIntersectWith(), 115
- implemented (nonabstract), 305
- IndexOf(), 107, 127
- IndexOfAny(), 127, 138
- Init(), 247, 249, 257
- Insert(), 145
- InstanceMethod(), 162
- Int32.TryParse(s, n), 131
- interfaces, 305, 308–309, 319–321
- IntersectWith(), 114
- introduction, 161–163
- InvokeMethod(), 195–196
- IsAllDigits(), 130–131, 133
- IsNullOrEmpty(), 128
- Join(), 133–134
- LastIndexOf(), 127
- Left(), 206–207
- List<T> class, 107–108
- Main(), 21, 135–136, 235, 257, 258, 276, 316–317
- naming conventions, 163, 175, 197
- nested loops, 86
- nonstatic (instance), 191–208, 357
- nonvoid compared to void, 186
- OrderBy<T>, 392, 393
- overloading inherited, 276–283
- Pad(), 134–136
- PadRight(), 135–136
- in query expressions, 373, 376, 391–397
- Read(), 154
- ReadLine(), 64, 123, 154
- Remove(), 145
- Replace(), 136–138
- Round(), 75, 226–228
- security, 220–224
- Select<T>, 392, 398
- Slice(), 207–208
- Sort(), 100, 317
- Split(), 131–133, 138–139
- String class, 127–129, 205–208
- Substring(), 126, 206–208
- SymmetricExceptWith(), 115
- ToArray(), 107
- ToBoolean(), 129
- ToDouble(), 129
- ToFloat(), 129
- ToInt32(), 120
- ToInt64(), 129
- ToLower(), 125–126
- ToString(), 126, 145, 291
- ToUpper(), 119, 125–126
- Trim(), 128–129, 134–136
- UnionWith(), 113, 115
- Update(), 180–181, 184
- value of, 170
- virtual, 288–290
- Where<T>, 392, 393, 396–397
- WriteLine(), 64, 109, 140, 189, 311
- Microsoft (Web site), 3, 8
- Microsoft.NET development, 13–14
- MixingStaticAndInstanceMethods program, 202–204

ModifyString program, 118–119
 module, definition, 223
 Mono, Novell (Web site), 1
 multiple arguments, passing, 172–173
 multiple dimensions, arrays compared to
 collection classes, 103
 multiple inheritance, C# restriction on, 307
 multiple variables. *See* arrays; collection
 classes

• N •

N-number specifier, Format() method,
 140
 namespaces, 21
 naming conventions
 classes, 151, 197
 constants, 69
 constructors, 234
 data members, 202
 errors in code, 406–407
 extended name in overloaded methods,
 276–277
 interfaces, 306
 methods, 163, 175, 197
 properties, 229
 regions, 18
 variables, 42, 101
 nested if statements, 67–69
 nested loops, 85–86
 .NET
 applications, 12
 development of, 13–14
 initiative of, 1
 .NET Common Language Runtime (CLR), 3
 .NET Framework, 13
 .NET Software Development Kit (SDK), 3
 nondeterministic destruction, 274
 nongeneric compared to generic
 collections, 116
 nonstatic data members, initializing, 240
 nonstatic (instance) methods
 compared to static, 162–163, 193–197,
 202–204
 current object accessing, 198–204
 extension type, 205–208
 Find() method as, 357

 passing objects to, 191–193
 nonvoid compared to void method, 186
 notational C#, 121
 Now property, DateTime, 42
 null items
 '0' (null) character, 38
 arrays and collections, 106
 event raising, 351
 IsNullOrEmpty () method, 128
 as operator, 264
 reference-type variables, 187–188
 string, 39
 numeric constants, declaring, 43–44
 numeric data types, logical comparison
 definition, 54

• O •

object-based compared to object-oriented
 languages, 286
 Object class
 accessing members of, 152
 collections problems with, 116
 definition, 151–152
 inheritance, 265–266, 291
 program based on, 153–154
 object-initializer syntax, 242–243
 object-oriented (OO) programming
 change management with interfaces,
 323–325
 overview, 209–215
 polymorphism's importance to, 286
 objects. *See also* classes; constructors;
 reference-type objects; variables
 access control, 152, 195, 198–204
 anonymous, 368
 compared to classes, 232
 current object, 196, 198–204
 discriminating between, 154–155
 initializing, 240, 368
 Observer, 347
 passing to methods, 191–193
 in static and nonstatic method mixing,
 204
 Observable object, 347
 Observer design pattern, 347–352
 Observer objects, 347

- OO (object-oriented) programming
 - change management with interfaces, 323–325
 - overview, 209–215
 - polymorphism’s importance to, 286
 - open-source software, definition, 1
 - operating systems, 1–2
 - operators
 - as, 263–265
 - arithmetic, 49–53, 120
 - assignment, 28, 51–52, 59, 155
 - binary compared to unary, 50
 - . (dot), 155
 - implicit promotion, 58
 - increment, 52–53, 84
 - is, 263–264, 287–288
 - lambda (\Rightarrow), 356
 - logical, 53–56, 120
 - query-related, 373, 374, 378–389, 392
 - string restrictions, 39
 - ternary, 67
 - working with types, 57–59
 - optional arguments (overloading), 174–177, 243–246, 276–283
 - order of arithmetic operators, 50
 - orderby operator, 373, 374, 392
 - OrderBy<T> method, 392, 393
 - out keyword, 179–180, 184–185
 - output, controlling, string, 134–139
 - OutputFormatControls program, 141–143
 - overloading, 174–177, 243–246, 276–283
 - override keyword, 288–290, 305
- *p* •
- Pad() method, 134–136
 - PadRight() method, 135–136
 - Parallel LINQ (PLINQ), 371
 - parameters, 362–363, 395–396. *See also* arguments, method
 - parentheses
 - in method names, 163
 - order of operators, 51
 - ParseSequenceWithSplit program, 131–132
 - parsing input, 128–130
 - Pascal-cased naming style, 42
 - PassByReference argument, 178–180
 - PassByValue argument, 177–188
 - PassObject program, 191–193
 - Pentium processor, floating-point calculations, 35
 - placeholder system, WriteLine() method, 140
 - PLINQ (Parallel LINQ), 371
 - PolymorphicInheritance program, 288–290
 - polymorphism
 - declared type, decision to use, 285–287
 - “do-to-each” loop trick, 291
 - interfaces, 322
 - introduction, 284–285
 - is operator and hidden methods, 287–288
 - overview, 215
 - virtual method and overriding, 288–290
 - Portable .NET (Web site), 1
 - postincrement operator, 52–53, 84
 - predefined delegates, 345
 - predefined interfaces, 310
 - Predicate<T> delegate, 357
 - preincrement operator, 52–53, 84
 - private keyword, 220–222, 293
 - procedural compared to conceptual view of OO programming, 210–213
 - procedures. *See* methods
 - Program.cs generation, 17
 - programming languages, introduction, 11–14
 - programs, introduction, 11–12, 17
 - project, definition, 16
 - projections on query input data, 379–383
 - Promote Local Variable to Parameter refactoring option, 167–168
 - properties. *See also* inheritance
 - in arrays and collection classes, 103–104
 - Count, 107, 109
 - data members, compared to, 229
 - definition of class, 159, 228–232
 - events, 350
 - interfaces, 319
 - Keys, 109
 - Length, 95–96
 - naming conventions, 229
 - Now property, DateTime, 42

query expressions, 379, 381–382
side effects, 230
static, 158–160, 230
property construct, definition, 228–232
protected keyword, 223, 273, 351
pseudocode, definition, 121
public class name modifier, 151
public keyword, 219–224, 293, 305
Publish/Subscribe design pattern, 347–352

• Q •

query expressions. *See also* lambda expressions
adding numbers array, 398
C# features using, 368
counting product, 398
definition and functions, 368–372
factorial computing, 399–400
filter expression, 377–379
inferring data types, 47
introduction, 2, 367
methods, 391–397
overview example, 372–376
reflection, 401
selecting and grouping results, 379–389
sorting results, 389–391
summing, 399
uses for, 376–377
queue, definition, 89
Queue<T> class, 104, 110

• R •

raising events, 349–351
range of digits, floating-point variable
compared to long, 35
ratios, integer's inability to handle, 32
reachable compared to unreachable
objects, 156
Read() method, 154
ReadLine() method, 64, 123, 154
readonly data member, 160
real compared to declared data type, 285
real numbers, 31–37
recursing, 282
ref keyword, 179–180, 184
Refactor Extract Method option, 167

refactoring, 163, 166–169
reference-type objects. *See also* string
variable type
assigning, 155–157
call-by-reference feature, 182–186
definition, 40
interface as, 308, 309–310
null and zero, 187–188
passing method arguments by, 178–181,
191–193
referencing classes to other classes,
157–158
returning to collection classes, 321
ReferencingThisExplicitly program,
201–202
reflection and query expressions, 401
#region/#endregion, 18
regions, code, introduction, 18
Remove() method, 145
RemoveWhiteSpace program, 137–139
Reorder Parameters option, 168
repeated instructions (loops). *See also*
foreach loop
breaking, 77–78, 85
“do-to-each” trick, 291
do...while, 76–77, 78–82
for, 82–84, 91–92, 96–97
infinite, 76, 84
introduction, 72–76
logical variable type, 73, 84
nesting, 85–86
scope rules, 82
string manipulation, 137–138
Replace() method, 136–138
resizing collections, 103
restriction (filter expression in queries),
373, 377–379
return statement
constructor restriction, 235
extended name, 276
interface as, 308
lambda expression, 364
method argument values, 182–186
nested loops, 86
query expressions, 375
reference types to collection classes, 321
reuse principle for programming, 254
Round() method, 75, 226–228

- rounding, integer, 32
- run-time compared to declared data type, 285
- S •
- SavingsAccount class, 258–260, 262–263, 273
- sbyte integer type, 30
- scope rules for loops, 82
- SDK (.NET Software Development Kit), 3
- sealed keyword, 300–301
- searching, strings, 127–128
- security (access control)
 - C# advantages, 13
 - class members, 219–228
 - classes' responsibility for, 213–214, 215
 - current object access, 198–204
 - delegates, 340
 - errors in code, 409–410
 - interface tool for, 319–321
 - keywords, 219–224, 340
 - methods, 220–224
 - object members, 152, 195
 - in object-oriented programming, 214, 215
- select operator, 373, 374, 379–389, 392
- selecting from query results, 379–389
- Select<T> method, 392, 398
- server, terminology issues, 327
- set section, 229, 231
- sets, 112–115
- shapes
 - collection class definition, 89
 - Dictionary<TKey, TValue> class, 108
- SharpDevelop (Web site), 3
- short-circuit evaluation operators, 56
- short integer type, 30
- side effects, properties with, 230
- signature, delegate, 363
- signed integer variables, 31
- Simonyi, Charles, 42
- simple operators, 49–50
- SimpleDelegateExample program, 337–339
- SimpleProgress program, 339–343
- SimpleSavingsAccount program, 255–257, 258
- Slice() method, 207–208
- Sort() method, 100, 317
- sorting
 - Array elements, 97–101
 - Compare() method, 121
 - query results, 374, 389–391
- SortInterface program, 310–316
- source files for executable programs, 12
- source program, creating, 15–18
- special characters, 38
- specifiers, 139–140, 305
- speed, calculation, floating-point variable limitations, 35
- Split() method, 131–133, 138–139
- SQL For Dummies (Taylor), 370
- stack, definition, 89
- Stack<T> class, 104, 110
- statement compared to expression lambda types, 362, 363–364
- static (class) methods. *See also* arguments, method
 - classes, relationship to, 161, 162, 191
 - compared to nonstatic, 162–163, 193–197, 202–204
 - definition, 161–163
 - with delegates, 343
 - example for reference, 163–169
 - Find() method as, 357
 - instance methods, compared to, 162
 - introduction, 161–163
 - nonvoid compared to void, 186
 - queries, 392
 - value of, 170
- static data members, initializing, 240
- static properties, 158–160, 230
- Strategy design pattern, 326–327
- String class
 - compared to string variable type, 117
 - extension methods for, 205–208
 - formatting strings, 139–143
 - functions of, 120
 - search methods, 127–128
 - trimming method, 128–129
- string variable type
 - array sorting, 97–101
 - case switching, 125–127
 - char, compared to, 40–41
 - command-line input, 128–134
 - comparing, 120–125

constants for, 44
 conversion restrictions, 45
 empty string, 39, 128, 187–188
 foreach loop, 126–127
 formatting, 139–143
 introduction, 39–40, 117–120
 List<T> class, 106
 logical operator restriction, 54
 in methods, 171
 Object class, 291
 objects, compared to, 187–188
 output, controlling, 134–139
 restriction on modifying, 118–119
 searching, 127–128
 StringBuilder class, 144–145
 switch statement, 71
 WriteLine() method, 189
 StringBuilder class, 144–145
 String.Empty value, 39
 Structured Query Language (SQL), 368,
 369–370, 371
 StudentClassWithMethods program,
 193–194
 Subclass class, 253, 266–270, 284
 subclasses, abstract class functions,
 299–300, 325
 subquery, 386
 subroutines. *See* methods
 substitutable classes, 262
 Substring() method, 126, 128, 206–208
 summing query results, 399
 Sun Microsystems, 13–14
 switch statement, 69–72, 86–87, 125–127
 SymmetricExceptWith() method, 115
 System.EventArgs class, 350
 System.MulticastDelegate class, 336

• T •

<T> notation, 104–105, 310, 357
 Taylor Allen G. (author), *SQL For Dummies*,
 370
 termination problems, 411–412
 ternary operator, 67
 text
 case in code, 19, 42, 86–87, 124, 125–127
 character variable types, 37–40, 44, 71,
 118. *See also* string variable type

third-party vendor languages (Web site), 14
 this keyword, 199–204, 248, 282–283
 time and date, variables for, 41–43
 TimeSpan class, 42
 Tittel, Ed (author), *XML For Dummies*, 370
 ToArray() method, 107
 ToBoolean() method, 129
 ToDouble() method, 129
 ToFloat() method, 129
 ToInt32() method, 120
 ToInt64() method, 129
 ToLower() method, 125–126
 Toolbox, saving text in, 22–24
 ToString() method, 126, 145, 291
 ToUpper() method, 119, 125–126
 Treat Warnings as Errors section, 281–282
 Trim() method, 128–129, 134–136
 TrimEnd() method, 129
 TrimFront() method, 129
 triple slashes, comments, 21
 true value for bool variable, 37
 truncation, integer, 32
 type, expression, introduction, 57–59

• U •

uint integer type, 30
 ULONG integer type, 30
 unary compared to binary operators, 50
 Unified Modeling Language (UML), 292, 293
 union operation, sets, 112
 UnionWith() method, 113, 115
 unreachable compared to reachable
 objects, 156
 unreachable memory blocks, 274
 unsigned int variable, 31, 44
 Update() method, 180–181, 184
 uppercase strings, 125–127
 usable interfaces in object-oriented
 programming, 213–214
 ushort integer type, 30
 using directive, basic console application,
 18, 21

• V •

value, passing method arguments by,
 177–180

- value-type variables. *See also* bool value-type variable; floating-point variable type; int (integer) value-type variable
 - basic method, 27–28
 - char, 38, 40–41, 44, 118
 - character, 37–40, 44, 71, 118
 - as compared to is operators with, 265
 - conversion of, 44–45
 - DateTime type, 41–43
 - decimal, 35–37, 45
 - definition, 40
 - double, 33–34, 35, 36, 44, 185
 - fractions, 31–37
 - inferring data types, 46–47
 - introduction, 27
 - in method arguments, 177–180
 - numeric constants, 43–44
 - real numbers, 32–35
 - var keyword
 - arrays, 102
 - function of, 368
 - inferring data types, 46–47, 363
 - queries, 372
 - in query expressions, 382
 - variable-length array, 92–95
 - VariableArrayAverage program, 92–95
 - variables
 - See also* classes
 - See also* collections
 - See also* reference-type objects
 - See also* value-type variables
 - definition, 22
 - grouping unlike-type, 149
 - as members of classes, 151
 - name errors, 406–407
 - type conversion errors, 407–409
 - VBScript, 46
 - VehicleDataOnly program, 153–150
 - virtual keyword, 288–290, 305, 351
 - virtual method, 288–290
 - Visual Basic, 12, 13, 14
 - Visual Basic .NET Database Programming For Dummies* (Mansfield), 370
 - Visual C#, 14–15
 - Visual C++, 14
 - Visual Studio, 1, 2, 4, 14, 15–19
 - void keyword, 185–186, 364
- *W* ●
- warnings, treating as errors, 281–282
 - where operator, 373, 378–379, 392
 - Where<T> method, 392, 393, 396–397
 - while loop, 72–76, 84
 - white space, trimming, 128–129
 - whole numbers, int limitation to, 30
 - WriteLine() method
 - collections, 109
 - if statement example, 64
 - interfaces, 311
 - placeholder system, 140
 - workings of, 189
- *X* ●
- x-hexadecimal specifier, Format() method, 140
 - XLINQ, 371
 - XML (Extensible Markup Language), 371
 - XML For Dummies* (Dykes and Tittel), 370
- *Y* ●
- yield statement, 375