

Index

Symbols and Numbers

wildcard, used in Like comparisons 162
 \$ (dollar sign), assertion made by 191
 & (ampersand) operator, using for string concatenation 167
 * (asterisk)
 as the greedy operator 162
 quantifier 185
 wildcard in Like comparisons 161
 [] (brackets), enclosing a character class 189
 ^ (caret)
 assertion made by 190
 placing before a character class 190
 {} (curly braces), surrounding the members of a set 268
 + (plus sign) quantifier 185
 << (left shift operator) 133–134
 >> (right shift operator) 133–134
 . (period) character class 187–188

? (question mark) quantifier 186, 187
 wildcard in Like comparisons 161
 () parentheses, surrounding regular expressions 191
 “80–20” rule 90

A

absorption law 270
 abstract methods 38, 41
 Add method 15, 41
 adding members to a set 271
 of the ArrayList class 59, 60, 111
 in a BucketHash class 215
 of the CollectionBase class 27
 for a dictionary object 201–202
 of the Hashtable class 219
 storing data in a collection 23
 for a strongly typed collection 42
 addEdge method 325
 AddRange method 59, 62
 addVertex method 325

- | | | | |
|--------------------------------|----------------|----------------------------|------------|
| Adelson-Velskii, G.M. | 298 | arrays | 15, 26, 46 |
| adjacency matrix | 323, 325 | building heaps | 288 |
| adjustShortPath | 344–346 | compared to ArrayLists | 65–70 |
| algorithms | | compared to BitArrays | 148 |
| advanced sorting | 283–296 | compared to linked lists | 228 |
| binary search | 93 | copying the contents of | |
| Dijkstra's algorithm | 340–350 | stacks into | 107 |
| greedy | 340, 352, 363 | declaring | 47, 48 |
| HeapSort | 288–293 | deleting elements from | 15 |
| iterative | 96 | designing hash table | |
| MergeSort | 285–288 | structures around | 210 |
| QuickSort | 283, 293–296 | doubling the number of | |
| recursive | 95–97, 285 | elements in | 58 |
| SelectionSort | 79 | dynamically resizing | 57 |
| ShellSort | 283–285 | finding prime numbers | 125 |
| shortest-path | 339 | initializing | 48 |
| sorting | 72–84, 283–296 | inserting elements into | 15, 68 |
| And method | 144 | in the knapsack | |
| And operator | 128 | program | 362 |
| AndAlso operator | 81 | median value in | 296 |
| anonymous groups | 191 | multidimensional | 52–54 |
| Append method | 137, 172 | problems with | 227 |
| application domain | 8 | re-dimensioning | 27 |
| arithmetic expression, storing | | searching for minimum and | |
| as a string | 104 | maximum values | 89 |
| Array class | | splitting into two halves | 294 |
| built-in binary search | | storing dynamic | |
| method | 97 | programming results | 354 |
| CreateInstance method | | storing linear collections | 15 |
| of | 47 | transferring the contents | |
| retrieving metadata | 50 | of Hashtable objects | |
| Array object | 47 | to | 222 |
| ArrayList class | 59, 60–65 | ASC function | 158 |
| ArrayList object | 101 | ASCII code | 158 |
| arraylists | 46 | ASCII values of a key | 211, 213 |
| as buckets | 215 | assertions | 190–191 |
| compared to BitArrays | 140 | “assignment” access | 42 |
| comparing to arrays | 65–70 | associations | 20 |
| contained by the | | associative set property | 269 |
| CollectionBase class | 41 | associative trays | 20 |

Index**383**

- asterisk (*)
 - as the greedy operator 162
 - quantifier 185
 - wildcard in Like
 - comparisons 161
- atEnd method 241
- averages, calculating in a
 - jagged array 57
- AVL trees 298
 - fundamentals 298
 - implementing 299–303
 - nodes in 299
- AVLTree class 299, 301–303
- B**
- \b assertion 191
- balanced binary trees 298
- base class 5, 6
- bases, numbers in other 108–110
- benchmark tests 6
- benchmarking. *See* timing tests
- binary number system 126–128
- binary numbers
 - changing the position of
 - bits in 133–134
 - combining with bitwise operators 129
 - comparing bit-by-bit 128
 - converting 108
 - converting integers into 134
 - converting to decimal equivalents 127
 - manipulating 128–130
- binary search 93–95
- binary search algorithm 93, 96–97
- binary search trees 21, 252
 - building 252–254
 - finding node and
 - minimum/maximum values in 257–259
 - handling unbalanced 298
 - inserting a series of numbers into 255
 - removing leaf nodes 259–261
 - traversing 254–257
- binary trees 21, 249, 250, 251, 366
- BinarySearch method 97
- BinarySearchTree (BST) class 252, 303
- binNumber array 143
- bins, queues representing 117
- binSearch function 94–95
- bit mask, 137. *See also* mask
- bit pattern for an integer value 134
- bit sets 124, 140
- bit shift demonstration
 - application 137–140
- bit values, retrieving 142
- BitArray class 124, 140
 - as the data structure to store set members 278
 - methods and properties 144
 - using 140–144
 - writing the sieve of Eratosthenes 145–148
- BitArray implementation 278–281
- BitArray method 148
- BitArray set 142, 279–281
- BitArrays, compared to
 - ArrayLists 140
- bitBuffer variable 137
- bits 126, 127
 - bitwise operators
 - working on 128
 - repositioning in binary numbers 133–134
 - representing sets of 124
 - setting to particular values 144
 - storing in a BitArray 141
 - working with sets of 140

BitSet array	143	carpet thief program	373, 376
BitShift operators	128, 133–134	CArray class	
bitwise operations	130	building	73–75
bitwise operators	128, 130–133	solving the sieve of	
black nodes in red-black		Eratosthenes	125
trees	304	storing numbers	75
Boolean truth tables	128	case-insensitive matching	197
Boolean values, computing		CCollection class	
the union of two sets	278	defining	26
brackets ([]), enclosing a		methods of	27–30
character class	189	modifying the heading	
breadth-first search	334–336	for	31
BSTs. <i>See</i> binary search trees		property of	27
bubble sort	75, 84	CEnumerator class	31–38
BubbleSort method	76, 77	CEnumerator object	31
bucket hashing	215–217	change, making	
BucketHash class	215–216	with coins	340, 363–365
buckets	215, 218	character array, instantiating a	
built-in data structure or		string from	152
algorithm	97	character classes	187–190
byte	127	characters	
Byte method	141	adding to the end of a	
Byte values	141	StringBuilder object	172
C		compressing a string of	366
Capacity property		defining a range of	188
of the ArrayList class	59	matching any in a string	187
of an ArrayList object	59, 61	removing from a	
of the StringBuilder		StringBuilder object	175
class	171	removing from either end	
Capture object	195	of a string	168
Captures property	195	replacing in a StringBuilder	
CapturesCollection		object	175
class	195–197	specifying a pattern based	
caret (^)		on a series of	187
assertion made by	190	Unicode values of	158
placing before a character		Chars method	113
class	190	Chars property	172
Carpet class	376	child, deleting a node with	
		one	261

Index**385**

child nodes in a binary tree	252	collections	14
children	250, 262–266	adding data to	23–25
Circle class	5	copying contents into an array	28
circular buffer	103	enumerating	25
circular linked list	233, 236–239	erasing the contents of	28
class method	157	iteration over	30
classes	1–6	looping through elements in	32
Clear method	15	retrieving items from a specified position	43
of the ArrayList class	59	returning the index of the position of an item in	29
of the CollectionBase class	28, 41	sub-categories of	15–21
of the CStack class	101	collisions	211, 215–218
for a dictionary object	201	comma-delimited strings	156
of the Hashtable class	222	comma-separated value strings (CSVs)	156
of the Stack class	107	commutative set property	269
Clear operation	100	Compare method	159
Clone method	144	CompareTo method	158, 159
code, timing	7	Compiled option for a regular expression	198
coin-changing problem	363–365	complete graph	321
Collection class		composition	5
built-in enumerator	25	compression of data	365–372
implementing using arrays	25–44	computer terms glossary, building	223–226
storing class objects	38–40	Concat method	167
collection classes		connected undirected graph	321
generic	23	connections in a network	336
introduced in VB.NET	44	constructor methods for	
collection operations	15	collection classes	27
collection properties	15	the CSet class	270
CollectionBase class		the CStack class	101
abstract and Public methods of	40	the Node class	230
building a strongly-typed collection	38	the Stack class	103
deriving a custom Collection class from	41–44	constructors	2
properties and methods of	26–29	for ArrayLists	60
		for BitArrays	140

- constructors (*cont.*)
- calling for the String class 151
 - creating StringBuilder objects 171
 - for an enumerator class 31
 - for strongly typed collections 42
- Contains method 15
- of the ArrayList class 59, 62
 - of the CollectionBase class 28, 41
 - of the Stack class 107
- ContainsKey method 222
- ContainsValue method 222
- continuous items 372
- ConvertBits function 137
- ConvertBits method 130
- CopyTo method of the
- ArrayList class 59
 - BitArray class 144
 - CollectionBase class 28, 41
 - DictionaryBase class 204
 - Hashtable class 222
 - Stack class 107
- cost. *See* weight of a vertex
- Count method 202
- Count property
- of an ArrayList 61
 - of the ArrayList class 59
 - of the CollectionBase class 27
 - of the CStack class 101
 - of the Hashtable class 222
 - retrieving collection items 24
 - for a stack 100
- CQueue class 111–112
- CreateInstance method 47
- CSet class 270
- BitArray implementation of 278
 - data member of 270
 - testing the implementation of 274–277
- CStack class 101–103
- CSVs (comma-separated value strings) 156
- CType function 38
- curly braces {}, surrounding the members of a set 268
- Current property of the CEnumerator class 32
- custom-built data structure or algorithm 97
- cycle 321
- D**
- \d character class 190
 - \D character class 190
- data
- adding to a collection 23–25
 - compressing 365–372
 - searching in a hash table 214
 - sorting with queues 116–119
- data fields 239
- data items, memory reserved for 8
- data members 2
- setting and retrieving values from 3
 - for a Timing class 10
- data structures, initializing to 3, 26
- data types
- determining for arrays 51
 - developing user-defined 2
 - Integer 17
 - native 151
 - numeric 17
 - setting for array elements 47
 - TimeSpan 10

Index**387**

- decimal numbers, converting
 to multiple bases 108–110
- default capacity, hash table
 with 218
- default constructor 2, 103
- definitions, retrieving from a
 hash table 223
- Delete method of the
 BinarySearchTree class 265
 SkipList class 316
- delVertex method 328
- DeMorgan's Laws 270
- depth of a tree 251
- depth-first search 331–333, 338
- DepthFirstSearch method 332
- Dequeue method, overriding 120
- Dequeue operation 19, 110
- derived class 5
- dictionary 20, 200
- DictionaryBase class 201–206
- DictionaryEntry array 204
- DictionaryEntry
 objects 201, 206, 223
- Difference method 273
- Difference operation 269, 278
- digraph 321
- Dijkstra, Edsger 340
- Dijkstra's algorithm 340–350
- direct access collections 15–18
- directed graph 321
- discrete items 373, 376
- DispArray subroutine 359
- displaying method 77
- displayNode method 252
- displayPaths method 346
- dispMask variable 137
- DistOriginal class for
 Dijkstra's algorithm 343
- distributive set property 269
- dollar sign (\$), assertion
 made by 191
- double hashing 217
- double quotation marks,
 enclosing string literals 150
- double rotation in an AVL tree 299
- doubly-linked list 233–236
- duration members of the
 Timing class 10
- dynamic arrays 15
- dynamic programming 352–363
- E**
- ECMAScript option for a
 regular expression 198
- edges 320
 adding to connect vertices 324
 nodes connected by 249
 representing a graph's 323
- Element member of the Node
 class 229
- elements
 accessing a
 multidimensional array's 53
 adding to an array 15
 inserting into a full array 67
 setting and accessing array 49
 sorting distant 283
- empty set 269
- empty string 151
- encapsulation 2
- EndsWith method of the
 String class 160
- Enqueue operation 19, 110
- EnsureCapacity method 172
- enumeration class, creating 30
- enumerator 25, 30–38
- Enumerator object for a hash
 table 220

388**INDEX**

- | | | | |
|---|----------|--|----------|
| equal sets | 269 | frequently searched items,
placing at the beginning | 90 |
| equalities for sets | 270 | full arrays, inserting elements
into | 67 |
| equality, testing for | 4 | functions, writing methods as | 4 |
| Equals method of the
CollectionBase class | 41 | | |
| String class | 158 | G | |
| Eratosthenes | 124 | garbage collection | 7, 8 |
| ExplicitCapture option for a
regular expression | 197 | garbage collector, calling | 8 |
| expression evaluator | 104 | GC object | 8 |
| extra connections in a
network | 336 | generalized indexed
collections | 20 |
| F | | generic collection class | 23 |
| Fibonacci numbers | 353–357 | genRandomLevel method | 316 |
| fields. <i>See</i> data members | | Get clause in a Property
method | 3 |
| FIFO (First-In, First-Out)
structures | 19, 110 | Get method | 142, 148 |
| FillSack method | 376 | getAdjUnvisitedVertex
method | 331 |
| finalizer methods | 8 | getCurrent method | 240 |
| Find method | 231, 258 | GetEnumerator method of the
ArrayList class | 59 |
| FindLast method | 235 | CollectionBase class | 41 |
| FindMax function | 90 | DictionaryBase class | 204 |
| FindMax method | 258 | IEnumerable interface | 30 |
| FindMin function | 89 | GetHashCode method | 41 |
| FindMin method | 258 | GetIndexofKey method | 208 |
| FindPrevious method | 232 | GetIndexofValue method | 208 |
| First-In, First-Out (FIFO)
structures | 19, 110 | GetLength method | 50 |
| fixed-length code | 366 | GetLowerBound method | 50 |
| For Each loop | 61 | getMin method | 344–346 |
| For Each statement | 25, 32 | GetRange method | 59, 64 |
| For loop | 24, 49 | GetSuccessor method | 264 |
| formatted string | 173 | GetType property | 50 |
| found item, swapping with
the preceding | 92 | GetUpperBound method | 50 |
| frequency of occurrence
for a character in a
string | 366 | GetValue method | 49 |
| | | global optimum | 352 |
| | | glossary, building with a hash
table | 223 |

Index**389**

- | | | | |
|----------------------------|---------------|------------------------------|------------|
| Graph class | | retrieving keys and values | |
| building | 322–325 | separately from | 219 |
| for Dijkstra's algorithm | 344 | searching for data in | 214 |
| preliminary definition of | 324 | hashing | 210–214 |
| graphs | 21, 320 | Hashtable class | 20, 270 |
| building | 323–325 | methods of | 221–223 |
| real world systems modeled | | in the .NET Framework | |
| by | 321 | library | 218 |
| searching | 330–336 | Hashtable objects | 218–219 |
| topological sorting | | header node | |
| application | 325–330 | in the LinkedList class | 231 |
| weighted | 340 | pointing to itself | 236 |
| greedy algorithms | 340, 352, 363 | resetting the iterator to | 241 |
| greedy behavior of | | header of a linked list | 228 |
| quantifiers | 186 | heading tag in HTML | 162 |
| “greedy” operator, * as | 162 | heap data | 8 |
| group collections | 21 | heap sort | 21 |
| grouping constructs | 191–194 | heaps | 8, 21, 288 |
| Groups method | 194 | building | 288–293 |
| growth factor for queues | 112 | removing nodes from | 291 |
| | | running all function | |
| H | | methods | 8 |
| handleReorient method | 311 | HeapSort algorithm | 288–293 |
| Hash function | | hierarchical collections | 20–21 |
| in a BucketHash class | 215 | hierarchical manner, storing | |
| for the CSet class | 271 | data in | 249 |
| hash functions | 20, 211–214 | Horner's rule | 213 |
| hash tables | 20, 210 | HTML formatting, stripping | 169 |
| adding elements to | 219 | HTML heading tag | 162 |
| building a glossary or | | Huffman, David | 365 |
| dictionary | 223 | Huffman coding | 365–372 |
| load factor of | 217 | HuffmanTree class | 370–371 |
| number of elements in | 222 | | |
| removing a single element | | I | |
| from | 222 | ICollection interface | 103 |
| removing all the elements | | IComparable interface | 299 |
| of | 222 | IDictionary interface | 201 |
| retrieving all the data | | IEnumerable interface | 30 |
| stored in | 221 | IEnumerator interface | 30 |

390**INDEX**

If-Then statement, short-circuiting	92	Insert method	15
IgnoreCase option for a regular expression	197	of the ArrayList class	60, 61
IgnorePatternWhiteSpace option for a regular expression	198	of the AVLTree class	301
immutable object	13	of the BinarySearchTree class	252–254
immutable String objects	170	of the CollectionBase class	41
immutable strings	16	for a doubly-linked list	234
increment sequence, sorting distinct elements	283	inserting a node into a heap	290
index		of the LinkedList class	231
of arrays	46	of the SkipList class	315
for the Remove method	43	of the String class	163
retrieving collection items	24	of the StringBuilder class	174–175
index-based access into a SortedList	208	InsertAfter method	240, 241
IndexOf method	15	InsertBefore method	240
of the ArrayList class	59, 62	InsertBeforeHeader Exception class	240
of the CollectionBase class	29, 41	InsertElement subroutine	69
of the String class	153	insertion	
infix arithmetic	104	into a linked list	229
inheritance	5	into a red-black tree	304
initial capacity for a hash table	218	Insertion sort	80–82
initial load factor for a hash table	218	improvement of	283
initialization list	48, 52	loops in	81
inner loop		speed of	84
in an Insertion sort	81	InsertRange method	60, 62
in the SelectionSort algorithm	79	Int function	75
InnerHashTable object	201	Int32 structure	17
inOrder method	255, 257	Integer array	55
inorder successor	262	Integer data type	17
inorder traversal	254, 255	integer index	15
		integer set members	278
		Integer variable	136
		integers	
		converting into binary numbers	134
		determining the bit pattern for	134

Index**391**

- integer-to-binary converter
 application 134–137
 intersection 21
 Intersection method 272
 Intersection operation 269, 278
 invalid index 43
 IP addresses, class storing 201
 isArray class method 51
 IsFull Private function 27
 IsMatch method 183
 Item method
 of the ArrayList class 60
 calling 101
 for a dictionary object 201–202
 of the Hashtable class 219, 220
 implementing as a Property
 method 42
 retrieving collection items 24
 retrieving values from a
 SortedList object 207
 items, retrieving from a
 collection 43
 iterative algorithm 96
 Iterator
 class 233, 239–241, 242–247
 iterFib function 354, 356
- J**
- jagged arrays 54–57
 Join method 156, 157
- K**
- Key property for a
 DictionaryEntry object 206
 key value, 251. *See also*
 key-value pairs.
 keys
 in a hash table 20, 210
 retrieving collection items 24
 retrieving hash table
 values 220
 storing along with
 collection data 23
 Keys method of the Hashtable
 class 219, 221
 key-value pairs. *See also* key
 value
 in a dictionary 20
 entering into a hash table 219
 in a priority queue 120
 removing from a SortedList 208
 storing data as 200
 Knapsack class 376
 knapsack
 problem 360–363, 372–376
 Knuth, Don 25
- L**
- label data member of the
 Vertex class 322
 Landis, E.M. 298
 Last-In, First-Out (LIFO)
 structures 19, 100
 lazy deletion 303
 lazy quantifier 187
 LCSubstring function 359
 leaf 251
 leaf node 259–261
 left bit shift 137
 left nodes of a binary
 tree 251
 left shift operator (<<) 133–134
 left-aligning a string 165
 Length method of the
 Array class 50, 58
 String class 153
 Length property of the
 StringBuilder class 171

- | | | | |
|------------------------------------|-----------------------|-------------------------------------|----------|
| levels | | load factor | 217 |
| breaking a tree down into | 251 | of Hashtable objects | 218 |
| determining for skip lists | 312 | initial for a hash table | 218 |
| of links | 311 | local optima | 352 |
| LIFO (Last-In, First-Out) | | logical operators | 128 |
| structures | 19, 100 | lookbehind assertions | 195 |
| Like operator | 161 | loops | 321 |
| linear collections | 14 | lowercase, converting strings | |
| linear lists | 18 | to | 168 |
| linear probing | 217 | M | |
| linear search, 29. <i>See also</i> | | machine code, translating | |
| sequential search. | | recursive code to | 353 |
| Link member of the Node | | MakeChange subroutine | 363–365 |
| class | 229 | mask, 134. <i>See also</i> bit mask | |
| linked lists | 227, 228, 311 | Match class | 182, 183 |
| inserting items into | 229 | MatchCollection object | 184 |
| marking the beginning | | matches | |
| of | 228 | at the beginning of a string | |
| modifying | 233–239 | or a line | 190 |
| object-oriented | | at the end of a line | 191 |
| design of | 229–233 | specifying a definite | |
| printing in reverse order | 235 | number of | 186 |
| referring to two or more | | specifying a minimum and | |
| positions in | 239 | a maximum number of | 186 |
| removing items from | 229 | specifying at word | |
| removing nodes from | 241 | boundaries | 191 |
| traversing backwards | 233 | storing multiple | 184 |
| LinkedList | | Matches class | 184 |
| class | 230, 236–239, 241–242 | MaxCapacity property | 171 |
| links | | maximum value | |
| adding to skip lists | 312 | finding in a BST | 258 |
| creating levels of | 311 | searching an array for | 90 |
| in a linked list | 228 | members | |
| probability distribution for | | removing from a set | 271 |
| levels | 314 | of a set | 268 |
| List, declaring as Protected | 41 | merge method, called by | |
| ListIter class | 239–241 | recMergeSort | 287–288 |
| list-oriented data structures | 99 | MergeSort algorithm | 285–288 |
| lists | 99 | | |

Index**393**

metacharacters	182	.NET environment, timing	
metadata, retrieving array	50	tests for	7–10
methods		networks	22
for collections	15	extra connections in	336
definitions allowing an		modeling with graphs	320
optional number of		study of	320
parameters	54	New, constructors named as	2
implementing topological		nextLink method	239
sorting	327–330	Node class	
performing operations on		for an AVL tree	
data in a structure	16	implementation	299–301
midpoint in a binary search	93	for a binary search tree	252
minimum spanning trees	336–339	building heap data from	289
minimum value		compared to the Vertex	
in a binary search tree	257	class	322
searching an array for	89	for Huffman coding	366
moveCol method	328	modifying for a	
MoveNext method	32	doubly-linked list	233
moveRow method	328	nodes	
multidimensional arrays	52–54	allocating to link levels	
counting the number of		randomly	312
elements in	50	connected by edges	249
performing calculations on		creating the linkage for	230
all elements of	53	deleting from a	
resizing	58	doubly-linked list	235
Multiline option for a regular		determining the proper	
expression	197	position for	253
MustInherit class	38, 201	inserting into a skip list	315
mutable object	13	inserting into linked lists	231
mutable String objects	170	inserting into the heap	
N		array	289
named groups	192–194	in linked lists	228
native data type	151	Node class data members of	229
negation of a character class	190	removing from BSTs	259–266
negative integers, binary		removing from heaps	291
representation of	135	removing from linked lists	232
negative lookahead assertion	195	returning the height of	299
negative lookbehind assertion	195	shifting in a heap	290
		of a tree collection	20

None option for a regular expression	197	ordered (sorted) arrays	227
nonlinear collections	14	ordered data for a binary search	93
non-numeric items, Set class for	268	ordered graph	321
noSuccessors method	327	ordered list	18
Not method of the BitArray class	144	OrElse operator	81
Nothing value at the end of linked list	228	organization chart	249
NP-complete problems	22	outer loop	
null object, equivalent of nullNode node in the RedBlack class	311	in an Insertion sort	79
NullReferenceException exception	49	in the SelectionSort algorithm	79
NUM.Vertices constant of the Graph class	325	Overrides modifier in the ToString method	4
numbers, primacy of	145		
numeric codes for characters	158	P	
numeric data types	17	PadLeft method	165
numeric values, Set class for	268	PadRight method	165
numVerts data member	325	pair, edges specified as a	320
		palindrome	102
O		ParamArray keyword	54
object-oriented programming (OOP)	1	parameter arrays	54
objects, converting to the proper type	38	parameterized constructor	2
octal, converting numbers from decimal to	108	parent	250
open addressing	217	parentheses (), surrounding	
operations, performed on sets	269	regular expressions	191
optimal solution for a greedy algorithm	363	Pareto, Vilfredo	91
Or method of the BitArray class	144	Pareto distributions	91
Or operator	129	Parse method	17
		partition value	296
		path	321
		finding the shortest in a graph	339–350
		in a tree	251
		Path method of the Graph class	344
		pattern matching	181
		patterns, comparing strings to	161
		pCount Protected variable	27

Index**395**

Peek method		PrintList method	232
of the CStack class	101	priority queues	19, 120
of the Stack class	106	Private access modifier	2
viewing the beginning item		Private class inside	
in a queue	110	CCollection	31
Peek operation	100	Private constructor for the	
period (.) character class	187–188	SkipList class	314
pig Latin	179	Private enumerator class	31
pIndex	26	probability distributions	91, 314
pivot value	296	process, picking the current	9
plus sign (+) quantifier	185	Process class	9
Point class	2	process handling	120
example constructors for	2	properties of sets	268, 269
example Property methods	3	Property methods	3, 6, 42
method to test for equality	4	Protected variable	
ToString method for	4	pCount	27
Pop method of the		pIndex	26
CStack class	101	Pugh, William	313
Stack class	104	punch cards, sorting	116
Pop operation	100	Push method of the	
position		CStack class	101
for items in a collection	24	Stack class	104
linear list items by	18	Push operation	19, 100
positional order within a			
collection	14	Q	
positive lookahead assertion	194	quadratic probing	217, 218
positive lookbehind assertion	195	quantifiers	185–187
postfix expression evaluator	123	quantity data, adding to a	
postOrder method	257	regular expression	185
postorder traversals	257	question mark (?)	
PQueue class	120–122	quantifier	186, 187
preOrder method	256	wildcard in Like	
preorder traversal	255, 256	comparisons	161
Preserve command	58	Queue objects	112
primary stack operations	104	queues	19, 99, 110
prime numbers		for breadth-first searches	334
as array sizes for hash		changing the growth factor	112
tables	211, 213	implementing using an	
finding	124	ArrayList	110

- queues (*cont.*)
 - operations involving 110
 - representing bins 117
 - sorting data with 116–119
 - specifying a different initial capacity 112
- QuickSort
 - algorithm 283, 293–296
- quotation marks, enclosing
 - string literals 150
- R**
- radix sort 116
- random number generator 74
- range operators in Like
 - comparisons 161
- ranges, adding to an ArrayList 62
- Rank property of the Array class 50
- ReadOnly method 43
- ReadOnly modifier 6
- read-only property 32
- real world systems, graphing 321
- recMergeSort recursive
 - subroutine 286
- recurFib function 353
- recursion
 - base case of 286
 - reverse of 352
- recursive algorithm 285
- recursive binary search
 - algorithm 95–97
- recursive call 354
- recursive code, translating to
 - machine code 353
- recursive method 301
- recursive process 294
- recursive program 353
- RedBlack class 306
- red-black trees 303
 - implementation code 306–311
 - inserting new items into 304
 - rules for 304
- ReDim command 57
- Redim Preserve command 69
- Redim Preserve statement 15, 74
- Redim statement 15
- reference types 7–10
- RegEx class 181, 182
- regular expressions 181
 - adding quantity data to 185
 - modifying using
 - assertions 190–191
 - options 197–198
 - surrounding in parentheses 191
 - working with 182–185
- Remove method 15
 - of the ArrayList class 60, 62
 - in a BucketHash class 215
 - of the CollectionBase class 41
 - for the CSet class 271
 - for a dictionary object 201–202
 - for a doubly-linked list 234
 - ensuring a legal index 43
 - of the Hashtable class 222
 - of the LinkedList class 232
 - removing data from a
 - heap 291, 293
 - of the String class 164
 - of the StringBuilder class 175
- RemoveAt method
 - of the ArrayList class 60, 62
 - calling 101
 - of the CollectionBase class 41
- Replace method
 - of the RegEx class 182, 184
 - of the String class 165
 - of the StringBuilder class 175

Index**397**

- Reset method of the
 - CEnumerator class 32
 - ListIter class 241
- Reverse method 60
- right nodes of a binary tree 251
- right rotation in an AVL tree 299
- right shift operator (>>) 133–134
- right-aligning a string 165
- RightToLeft option for a
 - regular expression 198
- Rnd function 75
- root node 21
 - of a binary search tree 252
 - colored black in a red-black tree 304
 - of a tree 250
- root of a subtree 251
- root sentinel node in the
 - RedBlack class 311
- rotation methods for the
 - AVLTree class 302
- rotations, performing in AVL trees 298
- RSort subroutine 119
- S**
- \S character class 190
- \s character class 190
- Search method of the SkipList
 - class 317
- search times, minimizing 90
- searching 86
 - advanced data structures and algorithms for 298
 - graphs 330–336
 - for minimum and maximum values 89–90
 - used by the IndexOf method 29
- Selection sort 79–80, 84
- SelectionSort algorithm 79
- self-organization of data 90
- separator for the Split
 - method 156
- SeqSearch function 86–87
- SeqSearch method 91–93
- sequential access
 - collections 18–20
- sequential
 - search 29, 86–87, 90–93
- Set as a reserved word 270
- Set class 268, 270–277
- Set clause in a Property
 - method 3
- Set method of the BitArray
 - class 144
- SetAll method of the BitArray
 - class 144
- sets 21, 268
 - adding members to 271
 - obtaining the difference of two 273
 - operations performed on 269
 - properties defined for 269
 - removing members from 271
- SetValue method 49, 53
- Shell, Donald 283
- ShellSort algorithm 283–285
- ShiftDown subroutine 69
- ShiftUp method 30, 290
- short-circuiting an
 - expression 81
- shortest path
 - Dijkstra's algorithm for determining 340–350
 - finding from one vertex to another 339–350
- shortest-path algorithm 339

- | | | | |
|---------------------------------|--------------------|-------------------------------|--------------|
| showArray method | | Split method | 156–158 |
| in the BubbleSort method | 78 | Stack class | 100, 103 |
| of the CArray class | 74 | stack objects | 103, 104 |
| showDistrib subroutine in the | | stack operations | 19, 104 |
| SimpleHash function | 213 | StackEmpty method | 100 |
| ShowString function | 359 | stacks | 8, 19, 99 |
| showVertex method | 325 | implementing without a | |
| Sieve of | | Stack class | 101–103 |
| Eratosthenes | 124, 145–148 | operations of | 100 |
| simulation studies, queues | | removing all items from | 107 |
| used in | 19 | specifying the initial | |
| single right rotation in an AVL | | capacity of | 104 |
| tree | 299 | in the TopSort method | 329 |
| Singleline option for a regular | | starting time, members of the | |
| expression | 198 | Timing class | 10 |
| Size method for the CSet | | starting time, storing | 9 |
| class | 271 | StartsWith method | 160 |
| skip lists | 298, 311 | static arrays | 15 |
| compared to linked | 311 | static method | 17 |
| determining levels for | 313 | String array | 156 |
| fundamentals | 311–313 | String class | |
| implementing | 313–318 | compared to | |
| performing deletions | | StringBuilder | 176–179 |
| in | 316 | methods of | 152–158 |
| SkipList class | 313–318 | string literals | 150 |
| SkipNode class | 313 | String objects | |
| Sort method | | comparing in VB.NET | 158 |
| of the ArrayList class | 60 | concatenating | 167 |
| in several .NET Framework | | creating | 151 |
| library classes | 297 | instantiating | 151 |
| SortedList class | 200, 206–208 | String processing | 181 |
| sorting | 72, 77–78, 116–119 | StringBuilder class | 13, 16, 150, |
| sorting | | | 170, 176–179 |
| algorithms | 72–84, 283–296 | StringBuilder objects | |
| spaces | | constructing | 171 |
| finding in strings | 153 | converting to strings | 176 |
| removing from either end | | modifying | 172–176 |
| of a string | 168 | obtaining and setting | |
| strings representing | 151 | information about | 171–172 |

Index**399**

- strings 16, 150
 - aligning 165
 - breaking into individual pieces of data 156
 - building from arrays 157
 - checking for palindromes 102
 - comparing to patterns 161
 - converting from lowercase to uppercase 168
 - defining a range of characters within 188
 - determining the length of 153
 - finding the longest
 - common 357–360
 - substring in 357–360
 - hash keys as 211
 - inserting into StringBuilder
 - objects 174
 - inserting into strings 163
 - matching any character in 187
 - matching words in 183
 - methods for
 - comparing 158–163
 - methods for
 - manipulating 163–170
 - replacing one with another 184
 - strongly connected directed graph 321
 - strongly-typed collection, building 38–44
 - structures 16–18
 - StudentColl class 43
 - subclass, deriving from the
 - CollectionBase class 41
 - suboptimal solution for a greedy algorithm 363
 - subroutines
 - timing 6
 - writing methods as 4
 - Subset method of the CSet
 - class 273
 - subset of another set 269
 - substring, finding the largest
 - common 357–360
 - Substring method 113, 153
 - subtrees, root of 251
 - Success property of the Match
 - class 183
 - successor, finding to a deleted node 263
 - swap code in the BubbleSort
 - algorithm 77
 - swap function 91
 - system time, assigning 7
- T**
- template for Property method
 - definition 3
 - test bed, examining sorting
 - algorithms 73
 - text file, reading in terms and definitions from 223
 - threads 9
 - three-dimensional arrays 52
 - TimeSpan data type 10
 - Timing class 10–12
 - comparing arrays to ArrayLists 65
 - comparing sorting algorithms 82–84
 - timing code, moving into a class 10–12
 - timing comparisons of the
 - basic sorting algorithms 82–84
 - timing tests 6
 - for the .NET environment 7–10
 - oversimplified example 6–7

400**INDEX**

- | | | | |
|--------------------------------|-------------------|-------------------------------|----------|
| ToArray method of the | | union | 21 |
| ArrayList class | 60, 64 | Union method of the CSet | |
| Stack class | 108 | class | 272 |
| ToLower method | 168 | Union operation | 269, 278 |
| top of a stack | 100 | universe | 269 |
| Top operation. <i>See</i> Peek | | unordered arrays, searching | 227 |
| operation | | unordered graph | 321 |
| topological | | unordered list | 18 |
| sorting | 325, 326, 327–330 | upper bound of an array | 50 |
| TopSort method | 328–330 | uppercase, converting strings | |
| ToString method | 4 | to | 168 |
| of the CollectionBase class | 41 | utility methods of the | |
| not available for objects | | Hashtable class | 221–223 |
| stored as Object type | 40 | | |
| of the StringBuilder class | 176 | V | |
| ToUpper method | 168 | Value property for a | |
| traffic flow, graphing | 321 | DictionaryEntry object | 206 |
| transportation systems, | | value types | 8 |
| graphing | 322 | values, retrieving based on | |
| transversal of a tree | 251 | keys | 220 |
| “Traveling Salesman” problem | 22 | Values method of the | |
| traversal methods with binary | | Hashtable class | 219 |
| search trees | 254–257 | variable-length code | 366 |
| tree collections | 20 | variables | |
| TreeList class | 368–369 | assigning the system time to | 7 |
| trees | 249 | stored on the heap | 8 |
| TrickleDown method | 291, 293 | stored on the stack | 8 |
| Trim method | 168–170 | Vertex class | |
| TrimEnd method | 168–170 | building | 322 |
| TrimToSize method | 60 | for Dijkstra’s algorithm | 343 |
| truth tables for bitwise | | vertices | |
| operators | 128 | building a list of | 323 |
| two-dimensional array | | in a graph | 320 |
| declaration | 52 | removing | 327 |
| storing results | 357 | representing | 322 |
| U | | W | |
| Unicode character set | 151 | \w character class | 190 |
| Unicode table | 158 | \W character class | 190 |

Index**401**

waiting lines, simulating	19	in regular expression	
wasVisited data member of		parlance	190
the Vertex class	322	returning from a hash	
weakly connected graph	321	table	223
weight of a vertex	320		
weighted graphs	340–350	X	
white space, excluding from a		x coordinate, storing	2
pattern	198	Xor method	144
wildcards in Like		Xor operator	129
comparisons	161	Y	
word boundaries, specifying		y coordinate, storing	2
matches at	191	Z	
words		zero base position in an array	46
matching in a string	183		
pulling out of a string	153		