

# Index

## • Symbols •

- (dash), using with method declarations, 146
- (decrement operators), 89–90
- ! (NOT) logical operator, 97
- != (not equal to) operator, 96
- % (modulus operator), 81
- % (percent character), using in `NSString`, 78
- %= compound assignment operator, 87
- & bitwise operator, 86
- && (logical AND) operator, 97
- += compound assignment operator, 87
- () (parentheses), using in operations, 82
- \* (asterisk), using in declarations, 98
- \*= compound assignment operator, 87
- . (dot notation)
  - using with object pointer, 313
  - using with data structures, 112
- // (slashes), using with comments, 85
- /= compound assignment operator, 87
- : (colon)
  - using with labels in `switch` statements, 208
  - using with methods and arguments, 147–148
- ;(semicolon)
  - checking for, 409
  - using with `for` loop, 213
  - using with statements, 43, 67, 73
- ? (conditional operator), 97–98
- @ (at) sign, meaning of, 44
- \ (backslash), using with string literals, 100, 102
- ^ bitwise operator, 86
- ^= compound assignment operator, 87
- { } (curly braces)
  - function action in, 43
  - using with blocks, 95
  - using with `struct`, 110
  - using with variables, 110
- | bitwise operator, 86
- || (logical OR) operator, 97
- |= compound assignment operator, 87
- } (closing brace)
  - terminating functions with, 123
  - using with instance variables, 145
- ~ bitwise operator, 87
- ++ (increment operators), 89–90
- += compound assignment operator, 87
- < (less than) operator, 96
- << bitwise operator, 87
- <<= compound assignment operator, 87
- <= (less than or equal to) operator, 96
- = (equal sign), using in declarations, 73
- = compound assignment operator, 87
- == (equality operator), 94, 96
- > (arrow pointer), 98, 131
- > (greater than) operator, 96
- >= (greater than or equal to) operator, 96
- >> bitwise operator, 87
- >>= compound assignment operator, 87
- " (double quote), using with string literals, 100
- , (comma)
  - operator for expressions, 90–91
  - separator for loops, 213

## • A •

- abstract class and superclass, 231
- accessors
  - naming, 322
  - using, 309, 313
  - using to get data from objects, 316–320
  - using with properties, 322
  - using within `Destination` class, 314–315
- action methods, declaring for buttons, 383–385
- ADC Reference Library, browsing, 50
- ADC Select membership, benefits of, 29
- addresses, existence in memory, 17
- `alloc` message, sending new message to, 169
- `alloc` method, using, 266, 283
- amount, storing in `Transaction` class, 204–205
- `anInteger` int, 74
- ANSI C, 58
- AppData file, locating, 334
- AppData property list, 324
- Apple Dev Center Web sites, 415
- Apple ID, creating and using, 22–24
- Apple Web site, look and feel of, 22, 32
- application-based data, 324. *See also* data
- applications, building and running, 79, 82, 156. *See also* iPhone application; programs
- `argc` variable, using with C array, 184

- argument count, using with C array, 184
  - argument names, using with messages, 147
  - argument type, including in method declaration, 146
  - argument vector, using with C array, 184
  - arguments
    - in `createBudget`: method, 150
    - passing transactions as, 226
    - passing variables to, 131
    - using with `country` string, 261–262
    - using with methods, 147–148
    - using `NSNumber` objects as, 171–173
  - `argv` variable, using with C array, 184
  - arithmetic operators. *See also* operators
    - modulus, 81
    - order of operands, 82
  - arrays. *See also* C arrays; fixed arrays; mutable arrays
    - adding objects to, 175
    - adding transactions to, 202, 212, 214
    - declaring, 350–351
    - versus dictionaries, 326
    - fast enumeration of, 176–177
    - multidimensional, 182–183
    - `NSArray`, 174–175
    - `NSMutableArray`, 174–175
    - objects in, 175
    - saving and initializing to `nil`, 350–352
    - saving balance data to, 352–354
    - static, 180–181
    - using indexes with, 175
    - using with property lists, 326
  - arrow pointer (`->`), 98, 132
  - ASCII codes, using, 100
  - assignment operator, 73, 78
  - asterisk (`*`), using in declarations, 98
  - at (`@`) sign, meaning of, 44
  - ATM transactions
    - completing implementation of, 362–366
    - creating, 365
    - processing, 359
  - Attributes Inspector, using, 375, 377, 397
  - `autorelease` pool allocation, adding, 320
  - `awakeFromNib` protocol method, using with Mac user interface, 401
- B •**
- backslash (`\`), using with string literals, 100, 102
  - balance data saved to array, 352–354
  - binary arithmetic, 86
  - bitwise operators, 86–87
  - blocks, defining, 95, 124
  - `BOOL`, 96
  - Boolean types, 93–96
  - break statement, using, 208, 220
  - breakpoints
    - activating and deactivating, 192
    - removing, 198
    - stopping, 197
    - using, 195–199, 411
    - verifying, 196
  - budget
    - managing, 106
    - revising, 130
  - `Budget` class, 140–141
    - class interfaces for, 144
    - modifying, 171–172
    - modifying for `Destination` class, 257–258
    - sending new message to, 154
  - budget instance variable, 227
  - budget member, changing, 136
  - `BudgetObject`, creating, 143
  - `BudgetObject.m` listing, 241–242
  - budget struct, 110–112
  - budget typedef, defining, 113
  - budget variable, changing, 197
  - `Budget.h` listing
    - accessors for data from objects, 318
    - `Budget` class, 258
    - returning back self, 272
  - `Budget.m` listing
    - accessors for data from objects, 316–317
    - `Budget` class, 257–258
    - plugging memory leaks, 290–291
    - returning back self, 272
  - bugs, fixing, 56. *See also* debugging tips
  - Build and Analyze feature, 199–200
  - Build and Debug button, 196
  - Build and Run button
    - location of, 79
    - using, 219
  - Build Results window, opening, 46, 187
  - built-in types
    - creating type names for, 113
    - `id` and `nil`, 166
  - bundles, placing plists in, 334
  - buttons
    - connecting, 389
    - connecting `IBActions` to, 402–405
    - declaring action methods for, 383–385
  - bytes
    - explained, 17
    - organization of memory into, 69



- C arrays. *See also* arrays
  - accessing elements of, 182
  - declaring, 182
  - main function, 183–184
  - multidimensional arrays, 182–183
- .c extension, explained, 39
- cash transaction
  - returning, 208
  - tracking, 109
- CashTransaction initializer listing, 267
- CashTransaction.h listing, 236, 273
- CashTransaction.m listing, 236, 259–260
  - plugging memory leaks, 292
  - returning back self, 273
- cast operator `()`, converting types with, 91
- categories, 356. *See also* methods
  - creating, 363–366
  - defined, 362
  - using, 366
  - using with useATM: method, 363
- CD for book
  - contents of, 53, 419–420
  - using, 420
- char data type, memory allocated to, 70–71
- char\*, 13
- characters, expressing with ASCII codes, 100
- charge transaction, tracking, 109
- chargeEuros function, calling, 118
- class code, division into two files, 158
- class definitions, 141, 229
- class diagram, 228–229
- class interfaces
  - for Budget class, 144
  - creating, 142–143
  - declaring, 143–144
  - entering `@end` compiler directive, 148
  - entering instance variables for, 145
  - entering methods for, 146–148
  - parts of, 144
- `@class` keyword, 234
- class names, case of, 144
- class versus instance methods, 153–154
- classes. *See also* container classes
  - abstract, 231
  - copying to projects, 379–380
  - creating interfaces for, 142–143
  - creating objects from, 141
  - evolution over time, 157
  - as extensions, 229
  - implementation part of, 142
  - importing header files for, 163–164
  - instance of, 141
  - interface part of, 142
  - modifying, 232
  - naming conventions, 165
  - versus objects, 140–141
  - selecting, 402–403
  - sending messages to, 153
  - specifying superclasses for, 160
  - using, 141
  - using multiple initializers with, 277
- Classes folder, creating, 159–160
- closing brace `()`
  - terminating functions with, 123
  - using with instance variables, 145
- Cocoa framework. *See also* design patterns
  - use of prefixes in, 103
  - using, 19–20
- code
  - adding for England, 156
  - analyzing, 199–200, 286–289
  - factoring, 120
  - reusability of, 62–63
  - reusing, 231–232
- code blocks, defining, 95, 124
- code libraries, use of, 39
- collections, using enumerations with, 176
- colon `:`
  - using with labels in `switch` statements, 208
  - using with methods and arguments, 147–148
- color picker, using in Interface Builder, 375–376
- comma `,`
  - operator for expressions, 90–91
  - separator for loops, 213
- Command Line Tool, choosing, 34
- comments, indicating, 85
- compiler
  - defined, 13
  - options for, 52–53
  - running instructions through, 15
- compiler directives, 68
  - `@end`, 152
  - `@implementation`, 149–150
- compiler warnings, paying attention to, 84, 410
- compilers, function of, 186
- composition technique, 247
- compound assignment operators, 87–88
- computer programming, process of, 9, 16
- computer programs. *See also* applications; projects
  - adding loops to, 216–219
  - building and running, 40–42
  - continuing to execution, 199
  - creating, 14–15

- computer programs (*continued*)
    - dividing into files, 158
    - extending, 156–157
    - extending functionality, 130–136
    - hiding internal mechanisms of, 60–61
    - I hate peanut butter and jelly example, 12–13
    - jumping to points in, 220
    - making enhanceable, 60
    - restarting from Debugger window, 199
    - running, 15–17
    - running in runtime environment, 19
    - terminating, 220
    - variables and instructions in, 12
    - writing, 10–11
  - condition, using with loop, 212
  - condition expression, using with Boolean types, 95
  - conditional operator (?), 97–98
  - Console, showing, 45, 192
  - const prefix, 101
  - constants
    - comparing expressions to, 209
    - declared, 101
    - #define, 101
    - using, 100
  - container classes, 174–176. *See also* classes
  - container property list objects, 325–326, 335
  - content engine, 246
  - continue control statement, using, 220
  - control objects, creating, 384. *See also* objects
  - control statements. *See also* statements;
    - switch statements
    - break, 220
    - continue, 220
    - exit, 220
    - goto, 220
    - return, 220
  - controller objects, 246
  - control-structure replacement. *See* polymorphism
  - copying classes to projects, 379–380
  - count method
    - of NSDictionary, 327
    - using with array, 176
    - using with for loop, 213
  - counter
    - declaring for loop, 212
    - varying transaction amount with, 218
  - Counterpart button, 165–166
  - counting, role in memory management, 281
  - country string, creating to use argument, 261–262
  - Cox, Brad, 58
  - .cpp extension, explained, 39
  - createBudget: method, defining, 150–151
  - credit card transaction
    - reporting, 155
    - for spend, 227
  - CreditCardTransaction.h, 236, 273
  - CreditCardTransaction.m, 236, 260, 273, 292–293
  - curly braces ({} )
    - function action in, 43
    - using with blocks, 95
    - using with struct, 110
    - using with variables, 110
  - currency symbols
    - adding to NSLog statements, 335
    - getting, 328, 331
  - customer care, 421
- D •
- dash (-), using with method declarations, 146
  - data. *See also* application-based data
    - accessing with pointers, 98–99
    - getting from objects, 307–308, 316–320
    - global accessibility, 137
    - providing for computers, 12
    - saving in separate file, 350–354
  - data structures
    - defining and declaring, 109–112
    - impact of changes on, 142
  - data types
    - char, 70
    - converting with cast operator (()), 91
    - creating with enumerations, 127–128
    - defined, 112–115
    - double, 70
    - float, 70
    - int, 70
    - signed and unsigned, 71–72
  - dealloc message
    - managing, 282–283
    - sending, 282
  - dealloc method
    - adding to Mac user interface, 402
    - invoking, 283
    - using, 289–290
  - deallocation process, tracing, 296

- Debugger
  - displaying local variables in, 193
  - showing, 192
  - using, 191–195, 411
  - viewing stack in, 193
  - viewing variables in, 193
- Debugger Console
  - displaying “Hello World” on, 42
  - displaying results in, 78
  - opening, 40–41, 46
  - paying attention to, 411
- Debugger window
  - Continue feature, 199
  - customizing look of, 194
  - Restart feature, 199
  - Step Out feature, 199
  - Step Over feature, 199
- debugging tips. *See also* bugs
  - checking for semicolons, 409
  - creating “paper” trail, 411–412
  - noticing compiler warnings, 410
  - noticing memory errors, 410
  - sending messages to `nil`, 411
  - testing incrementally, 412
  - thinking, 412
- decimals, declaring, 83–86
- declarations
  - defined, 74
  - specifying values in, 73
- declared properties, implementing, 311–312.
  - See also* properties
- decrement operators (`--`), 89–90
- `#define` constant, 101–102
- defined data types, 112–115
- delegate, example of, 381
- delegation
  - adding to transactions, 360–362
  - overview of, 356
- design patterns, 245. *See also* Cocoa
  - framework; MVC (Model-View Controller) pattern
    - delegation, 356
    - Target-Action, 385
- `Destination` class
  - adding Transaction object to, 362
  - declaring, 382
  - declaring methods for, 254
  - declaring properties for, 309–311
  - using accessors within, 314–315
- `Destination` methods, releasing, 295
- `Destination` object
  - adding, 253–254
  - creating, 261–262, 282
  - creating objects from, 284
  - initializing, 262
  - sending transaction amounts to, 262
  - sending transactions to, 386
- `@Destination` project, creating, 250–253
- `Destination.h` listing
  - destination design, 254–255
  - returning back self, 274
- `Destination.m` listing
  - adding `synthesize` to, 310–311
  - `Destination` class, 255–256
  - main in `Vacation.m`, 294–295
  - plugging memory leaks, 293–294
  - potential memory leak in, 287–288
  - returning back self, 274–275
- developer programs, types of, 22
- development advice
  - coding, 415, 417
  - finishing software, 417
  - garbage collection, 414
  - initialization, 415
  - linear approach, 416
  - managing objects, 413–414
  - managing `switch` statements, 414
  - memory management, 414
  - nonlinear approach, 416
  - using comments, 414
  - using documentation, 415
- development tools, Xcode and Interface Builder, 20
- dictionaries. *See also* keys; plists
  - adding entries to, 337–338, 341, 344
  - versus arrays, 326
  - assigning values to keys in, 341
  - creating, 327–329, 340–341
  - iterating through, 329
  - keys and values, 326–327
  - looking up values in, 328
  - mutable, 334–335, 344
  - property list objects in, 340
  - storing data in, 344–345
  - updating, 345–350
  - verifying, 343–344
  - writing to files, 346
- dictionary and plist modification, 346–348
- dictionary entries, hiding, 337–338
- dictionary of dictionaries
  - creating, 340
  - creating in `main`, 341
  - managing, 340–342
- digits, displaying, 86

- directives
  - compiler and preprocessor, 68
  - #define, 101
- do while loop
  - sequence of, 215
  - using, 215
  - versus while loop, 215
- documentation
  - accessing, 170
  - Find, 52–53
  - header file for symbols, 49–50
  - Help menu, 50
  - Quick Help, 49
- documentation window, features of, 50
- dollars, converting to foreign currency, 219
- dollars argument, 168
- dot notation (.)
  - using with object pointer, 313
  - using with data structures, 112
- double data type
  - displaying, 85
  - explained, 70
  - using, 168
- double dollars variable, calling, 123
- double object, 118, 123
  - replacing with NSNumber, 168, 171
- double quote ("), using with string literals, 100
- dynamic binding, 243

## ● E ●

- editor view, splitting, 165
- Editor window
  - displaying line numbers in, 187
  - error displayed in, 192
- else and if instructions versus switch statements, 208–209
- else structure, using with Boolean types, 95
- empty main function. *See also* main function for compound assignment operators, 87–88 for increment and decrement operators, 89 starting with, 75, 80–81
- encapsulation
  - explained, 58, 142
  - versus inheritance, 229
  - overview of, 59–61
  - versus polymorphism, 62
  - use of, 245
- @end directive
  - entering, 152
  - using with class interfaces, 143, 148
- England, adding code for, 156

- enhanceability, considering, 136–137
- enumerations (enum)
  - changing types of, 219
  - fast, 176
  - using to create data types, 127–128
  - using with collections, 176
- equal (=) sign, using in declarations, 73
- equality operator (==), 94, 96
- error: argument, using with plists, 351–352
- error messages, logging, 44
- error types
  - logic, 189–190
  - runtime, 188–189
  - syntax, 186–187
- europa struct budget, 112
- execution, continuing, 192
- exit control statement, using, 220
- expenses, tracking, 106
- expressions
  - comparing to constants, 209
  - constants as, 100
  - defined, 78
  - evaluating, 78
  - evaluating for switch statements, 207
  - for updating counters, 212
  - using comma operator (,) with, 90–91
- extensibility
  - considering, 136–137, 156–157
  - defined, 55
  - example, 381
- extensions, classes as, 229

## ● F ●

- %f string format specifier, 79, 85
- factoring code, 120
- fast enumeration, 176, 329
- file manager, using with plist, 351
- files
  - adding to projects, 160, 398
  - dividing programs into, 158
  - naming, 160–161
  - naming conventions, 165
  - placing in folders, 235, 363
  - saving, 388
  - saving data in, 350–354
  - switching between, 165
- Find and Replace, 380
- Find feature in Xcode, using, 52–53, 388
- First Program folder, opening, 39
- fixed arrays, 180–181. *See also* arrays; mutable arrays
- floating point numbers, 72, 86

floats, 70, 83–85, 121–122  
 turning strings into, 386  
 using with constants, 102  
 using with decimals, 83–86

folders  
 creating, 158–159  
 placing files in, 235, 363

for in statement, using, 202

for loop  
 versus for in, 213  
 execution flow, 211  
 using, 210–213  
 using with transactions, 242  
 versus for while loop, 214

foreign currency, converting US dollars to, 219

formatting options, setting, 46

Foundation framework, 42–43

frameworks, using, 19–20

function elements  
 functionArgument, 121  
 functionName, 121  
 returnType, 120–121

function prototypes, declaring, 128–129

function versus method declaration, 146

functionality  
 adding, 55  
 coding in main function, 152–156  
 extending for programs, 130–136  
 improving or changing, 56

functions. *See also* main function  
 calling, 124  
 defined, 43  
 defining blocks in, 124  
 ending, 220  
 enumerations, 127–128  
 execution, 122  
 explained, 18  
 including return statement in, 123–124  
 in instructions, 68  
 as methods, 141  
 versus methods, 147  
 in modules, 56–57  
 moving instructions into, 119  
 parts of, 122  
 passing to variables, 131  
 scope of variables, 124–126  
 statements as, 115  
 stepping into, 192  
 unions, 126–127  
 using modules as, 115

## • G •

garbage collection, 279, 301–303

GCC 4.2 - Warnings, customizing, 52

getter method, using, 309, 311

global variable scope, 125–126

goto control statement, using, 220

greater than (>) operator, 96

greater than or equal to (>=) operator, 96

Groups & Files view  
 External Frameworks and Libraries  
 folder, 39  
 First Program in, 38  
 Products folder, 39  
 Source folder, 39

## • H •

.h files, creating, 161, 165

header files  
 getting for symbols, 49–50  
 importing for class, 163–164

“Hello World,” displaying on Debugger  
 Console, 42, 44

Help menu search field, using, 50–51

## • I •

%i string format specifier, 79

IBActions  
 connecting to buttons, 402–405  
 versus IBOutlet, 384

IBOutlets  
 adding, 383  
 connecting in Mac user interface, 402–405  
 displaying, 389

id built-in type, 166

identifiers  
 defined, 70  
 following rules for, 72–73

Identity Inspector, opening, 402

if else construct  
 versus switch statements, 208–209  
 using with Boolean types, 95

if keyword, using with Boolean types, 94–95

if statement, 94–95

@implementation compiler directive, 149–150, 158

implementation files, looking at, 165

#import, 42, 75–76, 99, 111, 114, 119, 125

#include preprocessor directive, 68

- increment operators (++), 89–90
- index, using with array, 175
- inheritance
  - adding files for subclasses, 234–235
  - applying to reusable code, 231–232
  - capabilities of, 230
  - creating `Transaction` superclass, 232–234
  - versus encapsulation, 229
  - implementing subclasses, 235–237
  - modifying `main` for classes, 238–242
  - multiple, 229
  - overview of, 228–229
  - using to create protocols, 231
  - using with polymorphism, 243
- `init` abbreviation, using with initialization methods, 267
- `init` message, separating from new message, 169
- `init` method, invoking for superclasses, 267–270
- initialization
  - overview of, 73–74
  - process of, 267
- initialization function, creating, 150
- initialization methods, 265–266. *See also* methods
  - beginning, 267
  - invoking for superclasses, 269–270
  - using with `NSNumber` objects, 169
- initializers, using multiple per class, 277
- initializing instance variables, 270
- `initWithAmount` initializer, 267
- `initWithDouble:` message, sending, 169
- input
  - doing something with, 9
  - getting for programs, 9
- instance variables. *See also* variables
  - accessing, 321
  - accessing from within classes, 312–313
  - adding via inheritance, 230
  - defining, 141, 226
  - entering for class variables, 145
  - initializing, 270
  - naming, 322
  - scoping, 148–149
  - specifying for class interfaces, 143
- instance versus class methods, 153–154
- instructions, 209
  - ending, 73
  - functions in, 68
  - moving into functions, 119
  - objects in, 68
  - operators in, 68
  - `printf` example, 13
  - repeating with loops, 210
  - running through compiler, 15
  - skipping over, 199
  - statements as, 68
  - writing for computers, 12–14
- `int` data type
  - memory allocated to, 70–71
  - variations on, 71
- `int` method, calling with new message, 168–169
- Interface Builder. *See also* Mac user interface; UI (user interface)
  - color picker, 375–376
  - creating Mac user interface, 395–397
  - creating user interface, 371–379
  - Label item, 373
  - launching, 372, 395
  - Library window, 395
  - opening Library window in, 372
  - saving projects, 378
  - selecting View, 375
  - Text Field item, 373
  - using, 388–390
  - using with Mac user interface, 402–405
- `@interface` compiler directive, 143–144, 149, 158
- interface files, looking at, 165
- interface objects, customizing, 321. *See also* objects
- `@interface` statement, deleting, 237
- iPhone application. *See also* applications; UI (user interface)
  - adding methods, 385–388
  - adding outlets, 383
  - connecting button in, 389
  - creating project for, 370–371
  - developing, 370–371
  - displaying keyboard in, 376
  - displaying user interface for, 376, 378
  - implementing Target-Action pattern, 383–385
  - replacing `#import`, 380
  - touching Text Field in, 376
- iPhone developer
  - registering as, 22–26
  - Standard and Enterprise Programs, 26–27
- iPhone Simulator
  - installing projects on, 378
  - launching, 390
  - running iVacation in, 388–390
  - “is-a” relationship, 235
- iterator, using with `for` loop, 213

iVacation, running in simulator, 388–390  
 iVacationViewController, 381–382  
 ivars. *See also* variables  
   accessing, 321  
   accessing from within classes, 312–313  
   adding via inheritance, 230  
   defining, 141, 226  
   entering for class variables, 145  
   initializing, 270  
   naming, 322  
   scoping, 148–149  
   specifying for class interfaces, 143

## • J •

jump statements, using, 220. *See also* statements

## • K •

keyboard, displaying in iPhone application, 376–377  
 keyboard shortcuts  
   Attributes Inspector, 375, 397  
   Build and Run, 40, 82  
   copying classes to projects, 380  
   currency symbols, 328, 331  
   Find feature, 388  
   header files for symbols, 49  
   Identity Inspector, 402  
   New File dialog, 160, 363  
   new files, 253  
   New Project Assistant, 33  
   new projects, 370  
   Project Find window, 380  
   saving files, 388  
   Simulate Interface, 376  
   Xcode Console, 108  
   Xcode Debugger console, 40–41  
 keys. *See also* dictionaries  
   assigning values to, 341  
   creating for dictionaries, 328  
   looking for, 335  
   using with dictionaries, 326–327

## • L •

label and text fields, connecting in Mac UI, 402–405  
 Label item  
   dragging from Library window, 373–374  
   updating text in, 387

labels, using with `switch` statements, 208  
 languages. *See* programming languages  
 late binding, 243  
 less than (<) operator, 96  
 less than or equal to (<=) operator, 96  
 libraries, using, 19–20  
 Library window  
   dragging Label item from, 373–374  
   opening in Interface Builder, 372, 395  
 line numbers, displaying in Editor window, 187  
 literals, using, 100  
 local variables. *See also* variables  
   calling in `main`, 122  
   declaring in `main`, 153  
   displaying in Debugger, 193  
   double dollars, 123  
 logic, implementing with Boolean types, 93–96  
 logic errors, encountering, 189–190, 195  
 logical operators, 97  
 loop statements. *See also* statements  
   adding to `main` function, 216–219  
   adding to programs, 216–219  
   do while, 215  
   for, 210–213  
   while, 213–214  
 loops  
   leaving, 220  
   skipping rest of, 220

## • M •

.m extension, 39, 161, 165, 234  
 Mac Dev Center, accessing, 28–29  
 Mac user interface. *See also* Interface Builder; UI (user interface)  
   adding files to project, 398  
   adding methods, 400–402  
   adding Target-Action, 400–402  
   adding targets, 400–402  
   adding text field, 396  
   adding view controller class to, 399  
   connecting elements of, 402–405  
   creating project for, 393–395  
   implementing in code, 398–402  
   running `mVacation` on, 402–405  
 main function. *See also* empty main function; functions  
   adding code for England, 156  
   adding `switch` statements to, 216–219  
   arguments in, 124  
   in `Budget Object.m`, 178–179, 238–239

- main function (*continued*)
  - for C array, 183–184
  - calling local variables in, 122
  - coding for `Destination` class, 260–263
  - creating dictionary of dictionaries in, 341
  - declaring local variables, 153
  - declaring `vacationBudget` variable for, 116
  - declaring variables in, 118
  - deleting commented code in, 117
  - enabling for `Destination.h`, 255
  - example, 116
  - getting into, 249
  - getting out of, 248–249
  - instantiating object, 153–154
  - listing for MVC pattern, 248–249
  - modifying for `NSNumber`, 172–173
  - modifying in `Vacation.m`, 318–319
  - in `Program.m`, 43
  - requirement of, 116
  - sending messages to objects, 154–155
  - using `Transaction` class in, 238–242
  - in `Vacation.m` listing, 260–261, 276, 294–295
- `MainMenu.xib`, using with Mac user interface, 395
- members, using with struct, 110
- memory
  - finding things in, 17
  - measuring for variables, 91–92
  - organization into bytes, 69
- memory addresses, 98
- memory errors, noticing, 410
- memory leak
  - defined, 280
  - detecting, 286–289
- memory management, 280
  - `autorelease`, 297–299
  - `autorelease` pool, 299–301
  - objects in arrays, 297
  - rule of, 285
  - rules, 303–304
  - using reference counting, 281–285
- messages
  - receivers of, 153
  - sending to classes, 154
  - sending to methods in classes, 237
  - sending to objects, 141–142, 154–155, 411
  - syntax for, 153
  - using argument names with, 147
- method behavior, altering via inheritance, 230
- method declarations, entering for class interface, 146–148
- method versus function declaration, 146
- methods. *See also* categories; initialization methods; view controller method
  - adding for user interface, 385–388
  - adding to Mac user interface, 400–402
  - adding via inheritance, 230
  - arguments for, 147–148
  - coding, 149–152
  - continuing use of, 285
  - declaring for buttons, 383–385
  - declaring for `Transaction` class, 205
  - defining, 151–152
  - ending, 220
  - versus functions, 147
  - functions as, 141
  - implementing for `Destination.m`, 255–257
  - instance versus class, 153–154
  - operations as, 141
  - overriding, 230
  - releasing, 289
  - stepping into, 192
  - stepping out of, 199
- model objects, 246
- modules
  - behavior of, 61
  - dividing programs into, 56
  - examples of, 18
  - using like functions, 115
- modulus (%) operator, 81
- multidimensional arrays, 182–183
- multithreaded systems, using nonatomic with, 311–312
- mutable arrays. *See also* arrays; fixed arrays
  - adding, 177–180
  - allocating and initializing, 175
- mutable dictionary, creating, 334–335, 344
- `mVacation`, running on Mac, 402–405
- `mVacationController.h` file, adding to, 399–400
- MVC (Model-View Controller) pattern. *See also* design patterns; objects
  - advantage of, 247
  - implementing, 247–249
  - overview of, 246–247
- My First Program, using arithmetic operators with, 75–77

## • N •

- naming conventions, 165
- New File dialog, opening, 160
- new message
  - effect of, 168–169
  - sending to `alloc` message, 169
  - sending to `Budget` class, 154
  - separating from `init` message, 169
- New Project Assistant, 370
  - starting, 107
  - starting for Xcode, 33
- `nil` object
  - built-in type, 166
  - initializing array to, 350–352
  - returning for initialization method, 270
  - returning for instantiated object, 270
  - sending message to, 195, 411
- NOT (!) logical operator, 97
- not equal to (!=) operator, 96
- NS prefix, use in Cocoa, 103
- `NSArray`, 174–175, 181
- `NSDictionary` methods, 327, 329
- `NSLog` function
  - computation in, 79
  - use of, 44
  - using to display results, 78
- `NSLog` statements
  - adding currency symbols to, 335
  - benefits of, 411
- `NSMutableArray`, 174–175
- `NSNumber` object
  - Class Reference, 170
  - converting to transaction, 202
  - creating, 168–169
  - deleting for loop statement, 218
  - initialization methods for, 169
  - replacing with `double`, 171
  - using, 168
  - using as argument, 171–173
  - using instead of `double`, 168
  - using with mutable array, 179–180
  - values returned by, 169
- `NSObject` class, extending, 144–145
- `NSString` Cocoa object
  - % (percent) character in, 78
  - features of, 44
- numbers, storing in property lists, 325

## • O •

- `objc/objc.h` header file, contents of, 166
- object life cycle, 280–281
- object pointer, assigning, 313
- `objectAtIndex:` message, sending, 175–176
- `objectForKey` method of `NSDictionary`, 327–328
- Objective-C
  - case-sensitivity, 145
  - invention of, 58
  - overview of, 17–18
  - Version 2.0, 20
- object-oriented environments
  - parts of, 18
  - transaction and budget objects in, 57
- objects. *See also* control objects; interface objects; MVC (Model-View Controller) pattern; property list objects; transaction objects
  - adding to arrays, 175
  - allocating, 265–266
  - in arrays, 175
  - behavior of, 57–59
  - checking from Variables pane, 195
  - versus classes, 140–141
  - creating, 265–266
  - creating from classes, 141
  - creating with `alloc` method, 283
  - customizing, 402–403
  - getting data from, 307–308, 316–320
  - getting pointers for, 155
  - initializing, 155, 266–267
  - instance of, 141
  - instantiating in `main`, 153–154
  - in instructions, 68
  - passing self argument to, 155
  - reflecting for arrays, 219
  - releasing, 285, 289
  - retain counts for, 281–282
  - sending messages to, 141–142, 153–155, 411
  - sending release messages to, 282
  - serializable, 325
  - template for, 141
- objects assigned to properties, releasing, 313–315
- operations
  - as methods, 141
  - order of, 82
  - using parentheses (()) in, 82

operators. *See also* arithmetic operators

- = (equal) sign, 73
- bitwise, 86–87
- cast, 91
- comma, 90–91
- compound assignment, 87–88
- conditional, 97–98
- floats, 83–86
- increment and decrement, 89–90
- in instructions, 68
- logical, 97
- relational and equality, 96
- sizeof, 91–92
- use with statements, 18
- using, 74–75

options: argument, using with plists, 351

outlets

- adding to Mac user interface, 400–402
- adding to user interface, 383
- connecting to text fields, 388

output

- displaying, 40
- getting from programs, 12
- storing, 15

## ● p ●

@package directive, using with instance variables, 149

parentheses (), using in operations, 82

peanut butter and jelly sandwich program, 10–12

percent character (%), using in NSString, 78

plist and dictionary modification, 346–348

plist listing, 341–342

plists. *See also* dictionaries

- adding complexity to, 336–340
- adding entries to, 343–345
- adding to projects, 329–331
- placing in bundles, 334
- reading in, 340–341
- troubleshooting, 334
- usefulness of, 335
- using, 332–334
- using file manager with, 351

pointers

- accessing data with, 98–99, 131–132
- assigning values to, 341, 343
- dereferencing, 131
- getting for objects, 155

polymorphism

- example of, 226–228
- explained, 58
- implementation of, 411

versus inheritance, 231

overview of, 61–62

purpose of, 240

using with inheritance, 243

preferences, setting for Xcode, 45

prefixes, use of, 103

primitive property list objects, 325

printf example, 13

@private directive, using with instance variables, 148–149

procedural programming, 56–57

program architecture, 140

Program.m, 42–45

programming code, reading line by line, 12–13

programming languages, overview of, 18–19

programming test, 10

programs. *See also* applications

adding loops to, 216–219

building and running, 40–42

continuing to execution, 199

creating, 14–15

dividing into files, 158

extending, 156–157

extending functionality, 130–136

hiding internal mechanisms of, 60–61

I hate peanut butter and jelly example, 12–13

jumping to points in, 220

making enhanceable, 60

restarting from Debugger window, 199

running, 15–17

running in runtime environment, 19

terminating, 220

variables and instructions in, 12

writing, 10–11

Project Find window, bringing up, 380

Project window, 36

projects. *See also* computer programs;

Xcode project

adding files to, 160, 398

adding plists to, 329–331

choosing from New Project window, 33–34

copying classes to, 379–380

creating @Destination, 250–253

creating for Mac user interface, 393–395

creating iPhone application, 370–371

installing on iPhone Simulator, 378

naming, 35

starting, 107

properties. *See also* declared properties

accessing via properties, 322

adding, 309–311

adding to Destination class, 309–311

assigning for user interface, 386

- naming conventions, 322
- proper use of, 320–322
- @property declaration, using, 309–310
- property list files. *See* plists
- property list objects. *See also* objects
  - containers, 325–326, 335
  - dictionaries, 326–329
  - in dictionaries, 340
  - primitives, 325
- property lists
  - defining, 324–325
  - using, 325–326
  - using with arrays, 326
- @protected directive, using with instance variables, 148
- protocols
  - adopting, 358–360
  - declaring, 357–358
  - defining, 367
- @public directive, using with instance variables, 148

## • Q •

Quick Help window, features of, 49

## • R •

- readwrite property attribute, using, 311
- red file name, meaning of, 39
- reference counting, using in memory management, 281–285
- relational operators, 96
- release message
  - versus retain, 285
  - sending, 175, 282–283
- reserved words, 102
- retain, using with declared properties, 311–312
- retain counts
  - decrementing, 283
  - determining, 281–282
  - incrementing, 283
  - versus release, 285
  - setting for Destination object, 284
- retain message
  - receiving, 175
  - releasing, 313
  - sending automatically, 313
  - using, 312–313
- return statement, 45
  - including in functions, 123–124
  - using, 220

- return type, 124, 146
- reusing code, 62–63, 231–232
- runtime environment, running programs
  - in, 19
- runtime errors, encountering, 188–189

## • S •

- saving
  - balance data to array, 352–354
  - data in separate file, 350–354
  - files, 388
  - projects in Interface Builder, 378
- scope
  - of instance variables, 148–149
  - of variables, 124–126
- SDK (software development kit)
  - choosing, 40
  - contents of, 21
  - downloading, 30–32
- self argument
  - passing to objects, 155
  - returning back, 271–277
  - using to send message to superclass, 268
- semantics, 69
- semicolon (;)
  - checking for, 409
  - use with statements, 43, 67, 73
  - using with for loop, 213
- sender argument, using with button, 385
- serializable objects, 325
- setter and getter methods, 309
- setter method, calling, 309, 313, 386
- signed and unsigned data types, 71–72
- Simulator. *See* iPhone Simulator
- sizeof operator, 91–92
- slashes (//), using with comments, 85
- spend method
  - emptying, 234
  - implementing, 227
- spendDollar: message
  - creating array for, 175
  - eliminating, 175
  - in mutable-array example, 180
  - sending, 202
- spendDollars function
  - calling, 118, 122
  - declaring, 120
  - terminating, 123
- spendDollars: message, 168
- spendDollars: method, implementing, 172–173
- stack, viewing in Debugger, 193

- statements. *See also* jump statements; loop statements; switch statements
    - in blocks, 95
    - declarations, 68
    - for displaying “Hello World,” 44
    - executing once, 212
    - as functions, 115
    - identifying, 43
    - including in blocks, 95, 124
    - instructions, 68
    - keeping on one line, 75
    - recognizing, 67
    - on two lines, 44
    - writing programs as series of, 18
  - Static Analyzer, using, 199–200, 286–289
  - Step into feature, 192
  - Step Out of feature, 192, 199
  - Step Over feature, 192, 199
  - string format specifiers, 78–79
  - string literal, 100
  - strings. *See also* symbol string
    - defined, 44
    - entering into dictionaries, 338
    - initializing, 261–262
    - keeping on one line, 75
    - storing in property lists, 325
    - theSandwich example, 13
    - turning into floats, 386
  - struct budget, defining with three variables, 114
  - struct data structure, 110
    - avoiding in declarations, 112–115
    - changing, 136, 140
    - creating from class objects, 141
    - defining, 113
    - passing as argument to function, 131
    - versus union, 126
  - structures. *See* data structures
  - sub items, hidden, 337–338
  - subclasses
    - adding files for, 234–235
    - capabilities of, 230
    - implementing, 235–237
    - versus superclasses, 228–229
  - subgroups, creating, 39
  - superclasses
    - invoking `init` method, 267–270
    - invoking initialization methods for, 269–270
    - specifying for class, 160
    - specifying for class interface, 143
    - versus subclasses, 228–229
  - switch statements. *See also* control statements; statements
    - adding to `main` function, 216–219
    - form of, 207
    - versus if else construct, 208–209
    - managing, 414
    - problems with, 220–221
    - sequence of, 207
    - using, 206–210
    - using labels with, 208
  - switch structure, 226
  - symbol string. *See also* strings
    - creating, 335–336
    - initializing, 335–336
    - using, 335–336
  - symbols
    - getting Quick Help for, 49
    - header files for, 49–50
  - syntax, 69
  - syntax errors, catching, 186–188
  - @synthesize statement
    - adding for delegation, 362
    - adding to `Destination.m`, 310–311
  - system requirements, 420
- T ●
- tab width, setting, 46
  - Target-Action pattern
    - applying to Mac user interface, 400–402
    - implementing, 383–385
  - tasks, breaking into modules, 15
  - templates, selecting, 34
  - text and label fields, connecting in Mac UI, 402–405
  - Text Editor
    - Bookmarks menu, 48
    - Breakpoints menu, 48
    - Class Hierarchy menu, 48
    - Code Folding, 48
    - Code Sense, 47–48
    - Counterpart button, 48
    - displaying tooltips in, 48
    - Included Files menu, 48
    - launching files in separate windows, 48
    - navigation bar, 48, 164
  - Text Field item, using in Interface Builder, 373
  - text fields
    - adding to Mac user interface, 396
    - connecting outlets to, 388
  - time stamp, example of, 44
  - tooltips, displaying, 48, 192

Touch Up Inside event, 389  
 trackSpending method, adding, 234  
 Transaction class  
   adding, 203–206  
   adding implementation for, 205–206  
   adding interface for, 204–205  
   creating protocol with, 231  
   using in main, 238–242  
   using with for loop statement, 217  
 transaction initializer listing, 268–269  
 transaction objects. *See also* objects  
   creating, 202  
   creating for switch statement, 209  
   managing, 203–206  
   processing, 208  
   removing UI type functionality from, 258  
   types of, 226, 355  
 Transaction superclass, creating, 232–234  
 transaction type, determining, 204–205  
 Transaction.h, 233, 259  
   adding delegation to, 360–361  
   opening in new window, 204  
   returning back self, 272  
 Transaction.m listing, 232–233, 259  
   plugging memory leaks, 291  
   returning back self, 272–273  
 transactions  
   adding delegation to, 360–362  
   adding to arrays, 212, 214  
   creating, 202  
   passing as arguments, 226  
   relating to budgets, 227  
   representing with double type, 168  
   sending to Destination object, 386  
   simulating, 133  
 transactions array, declaring, 218  
 transactionType, using switch statement  
   with, 206  
 transparency, defined, 56  
 triangles, pointing down, 337–338  
 typedef keyword, using with data types,  
   112–114, 116  
 typos, causing logic errors with, 190

## • U •

%u string format specifier, 79  
 UI (user interface). *See also* Interface Builder;  
   iPhone application; Mac user interface  
   adding methods for, 385–388  
   creating, 371–379  
   displaying for iPhone, 376, 378  
   example of, 397

UI behavior, simulating, 262  
 UI type functionality, removing from  
   Transaction objects, 258  
 UIApplicationDelegate Protocol, 381  
 UML (Unified Modeling Language) notation,  
   228–229  
 union function  
   versus struct, 126  
   using, 127  
 uPhone  
   extensibility of, 61–62  
   invention of, 59–60  
 US dollars, converting to foreign currency,  
   219  
 useATM: method, adding, 363–366

## • V •

Vacation Budget project, creating, 107  
 vacationBudget England variable,  
   updating, 130  
 Vacation.m, main function in, 276, 318–319  
 valueForKey: method, 335  
 values  
   accessing in addresses, 99  
   assigning, 75–77  
   assigning to pointers, 341, 343  
   checking from Variables pane, 194–195  
   looking up in dictionaries, 328  
   specifying in declarations, 73  
   using with dictionaries, 326–327  
 variables. *See also* instance variables; local  
   variables  
   Boolean types, 93–96  
   changing, 197  
   declaring, 70, 73, 75–77  
   defined, 69  
   descriptiveness of, 77  
   examples of, 13  
   measuring memory used by, 91–92  
   passing functions to, 131  
   passing to arguments, 131  
   scope of, 124–126  
   setting watch points on, 197–198  
   versus structure type names, 112  
   using with decimals, 83–86  
   using with struct, 110  
   viewing in Debugger, 193–194  
   Variables pane, using, 194–195  
 verification code, entering as iPhone  
   developer, 24–25  
 view controller class, adding to Mac user  
   interface, 399

view controller method, overriding, 387. *See also* methods  
 view objects, 246  
 virtual memory, 281  
 void, 116–117, 120–121, 123  
   including in argument list, 121  
   return type, 122

## • W •

watch points, setting on variables, 197–198  
 while loop  
   versus for loop, 214  
   sequence of, 214  
   using, 213  
   using with transactions, 242  
 while loop, using, 213–214  
 Wiley Product Technical Support, 421

## • X •

Xcode, setting preferences for, 45–47  
 Xcode 3.2 developer tools, using, 20  
 Xcode Console, opening, 108  
 Xcode Debugger Console. *See* Debugger Console  
 Xcode Organizer, 33

Xcode Preferences window  
   closing, 46  
   opening, 45

Xcode project. *See also* projects  
   creating, 32–36  
   Detail view, 36–37  
   Editor view, 36–38  
   Groups & Files list, 36  
   launching, 32  
   status bar, 36, 38  
   Text Editor navigation bar, 36–37  
   toolbar, 36–37

Xcode Text Editor  
   Bookmarks menu, 48  
   Breakpoints menu, 48  
   Class Hierarchy menu, 48  
   Code Folding, 48  
   Code Sense, 47–48  
   Counterpart button, 48  
   displaying tooltips in, 48  
   Included Files menu, 48  
   launching files in separate windows, 48  
   navigation bar, 48, 164  
 Xcode window, activating, 378, 397

## • Z •

zero, dividing by, 191–192