30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 469

Index

#define, and symbolic names, 122
#include
  <ctype.h>, 178–179, 455
  <fstream>, 198, 201
  iostream, 16, 57, 201
  math.h (math library), 16, 57–58, 455–456
  stdafx.h (Visual Studio), 14
  stdlib.h (standard library), 175–177, 278, 457
  string (**string** class), 187, 190
  <string.h> (C string library), 170
  <time.h>, 108, 457
  angle brackets for standard libraries, 283
  function declarations, 82–83
  multi-module projects, 231–232
  preprocessor directives, 16–17
  quotation marks for project files, 283
% (remainder function). *See* remainder (%) function
& address operator. *See* address operator (&)
&& (AND Boolean operation), 54
* (indirection / at operator). *See* indirection operator (*)
*= (multiplication-assignment operator), 265. *See also* assignment operators
// (comments), 24–25, 49
~ (in class destructors), 364
:: (in scope prefix). *See* scope prefix (::)
\ (backslash / escape), 172, 200
\0 (null character), 166, 168
\b (backspace), 172
\n (newline), 172. *See also* newlines
\t (tab), 172
|| (OR Boolean operation), 54
+ (addition operator), and string concatenation, 188–189
++ (increment operator)
  and array pointers, 155, 159
  introduced, 51–52
  and pointers, 142
+= (addition-assignment operator), 91. *See also* assignment operators
<,> (angle brackets) for standard libraries, 283
<> stream operator. *See* stream operators
<= less than or equal to operator, 49
<...> libraries and headers. *See* **#include**
= (assignment). *See* assignment (=)
== (equality). *See* equality (==)
-> (dereference operator), 346–347
>> stream operator. *See* stream operators

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 470

. (dereference operator), 346–347
" (double quotation mark)
  project file includes, 283
  and strings vs. characters, 180
! (NOT Boolean operation), 54
; (semicolon). *See* semicolons
' (single quotation mark), and strings vs. characters, 180

absolute value function (**abs**), 275, 278, 456
abstract classes. *See* interfaces
access levels (class data) table, 406
acos function, 456
addition operator (+), and string concatenation, 188–189
addition-assignment operator (+=), 91
address expressions, scaling, 155
address operator (&)
  and array elements, 149, 152
  introduced, 142
  and passing variable references, 146–147
  and reference variables, 302–304
  and swap function, 148
addresses, memory, 140–142
advancing to next print line. *See* newlines
aggregates, array initialization, 113
allocation, memory. *See* **new** operator
AND Boolean operation (&&), 54
angle brackets for standard libraries, 283
anonymous variables, 344
ANSI C++ data types, 440
answers to exercises, on CD, 11
applications, defined, 4
argc, 220, 223
arguments
  command line, 219–223
  empty list, 85
  to functions, 85, 86
  as local values, 100–101
  main function, 220, 223
  object initialization, 219–221, 249
argv, 220, 223
arithmetic operations, on pointers, 154–155
array elements examples, 115–116, 117–121
array sorting example, 149–153
arrays
  address of, 154–155, 159, 193
  and aggregates, 113
  declaring, 112
  indexing. *See* indexes, array
  initializing, 113, 136–137
  introduced, 111–113
  passing to functions, 158–159
  pointer usage, 156–157
  strings, 122–123. *See also* string data
  of strings, example, 181–185
  two-dimensional, 136–137
  zeroing via pointers, 156–160
Artificial Intelligence, and computer decision-making, 37
ASCII code
  and binary files, 206–208
  and characters, 20
  and string data, 163–164, 180
  table, 451–452
asin function, 456
assign function, of **string** class, 193
assignment (=)
  and data-casting, 88
  and equality (==), 38, 40–41
  and expressions, 53
assignment functions, and subclassing, 396

assignment operator functions
  and copy constructors, 329–330
  described, 329–330
  and **this** pointers, 374–375
assignment operators. *See also* assignment operator functions
  addition-assignment operator (+=), 91
  multiplication-assignment operator (*=), 265
  string data, 188–189
  table, 91
association
  C++ operators, 435–437
  and pointer arithmetic, 159–160
at operator (*). *See* indirection operator (*)
atan function, 456
atof function, 175–177, 454
atoi function, 175–177, 454
automatic functions. *See* compiler-supplied

backslash (\). *See* escape sequences
backspace (\b), 172
base classes
  passing subclass types, 427
  and **public** keyword, 388–389
  and subclasses, 385
Basic (computer language), 6
Beatles example, 183
binary data, storing dollars and cents, 268
binary files
  described, 206–208
  examples, 211–216
  fields, 209
  read and **write** functions, 207, 208–210
  records, 213
  reinterpret_cast, 210
  seekp function, 216
binary numbering system 140–141
bitwise operators, confusion with Boolean operators, 55
bool data type, 51, 331–332
Boolean algebra, 54
Boolean operators
  AND (&&), 54
  bool data type, 51, 331–332
  confusion with bitwise operators, 55
  example, 56–57
  NOT (!), 54
  OR (||), 54
  precedence, 55
  short-circuit logic, 54–55
  table, 54
  and true / false, 50
bounds checking
  card shuffling example, 134
  dynamic strings, 360, 364
  importance, 135–136
  strings, 167
break statement
  and infinite loops, 77
  introduced, 61
  in **switch-case** statements, 231
  usage and syntax, 445
buckets (variables), 33
building a C++ program
  described, 9–10
  example, 12–15
  in RHIDE, 12–13
  in Visual Studio, 13–14
butterfly effect, 108

C (computer language)
  data casting, 36–37
  design goals, 6
  and object-orientation, 251, 254–257
  structures and classes, 263

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 472

C# (computer language), 7
C++ language
  compared to other languages, 6–8
  as high-level language, 6
  and object-oriented features, 7–8, 245
  operators, 435–437
  syntax summary, 441–449
C++ library
  and **#include**, 74
  common functions, 453–457
  and linking, 9, 31, 234
C++ programs
  building, 8–11, 12–14
  comments, 24–25, 49
  compiling, 9–10
  integrated development environments, 8
  linking, 9–10
cards, dealing examples, 123–129, 130–135
case statements, 230–231
case-sensitivity, 12
casting data. *See* conversions between data types
catch statements. *See* exception handling
CD (included with this book), xxv, 11
ceil function, 456
Celsius conversions, 23–27
cents, binary precision, 268
char data type. *See* data types, **char**
char* notation, 123, 186
characters vs. strings, 177–179, 180, 181
cin
  and >> stream operator, 173
  as data object, 20
  default values, 174
  getline function, 168–171, 175–177
  reading string data, 172–174
  string data, 189
  and white space, 173–174
class data. *See* data members
class keyword, compared to **struct**, 263
classes. *See also* objects
  abstract. *See* interfaces
  accessing private data, 262–263
  assignment functions. *See* assignment (=)
  assignment operator functions. *See* assignment operator functions
  as C++ language extensions, 268
  constructors. *See* constructors
  data members. *See* data members
  declaration, 261–262, 448–449
  defined, 248–249
  destructors. *See* destructors
  and encapsulation, 246–247
  initializing. *See* constructors
  inlining, 271–273, 292–293
  interfaces. *See* interfaces
  member functions. *See* member functions
  namespaces, 250
  operator functions. *See* operator functions
  test of equality (==) function, 330–331
cleanup, at object destruction. *See* destructors
closing, file streams, 199
code, 4
comma-delimited input example, 181–185
command line arguments, 219–223
commas, in numbers, 59–60
comments, 24–25, 49
comparisons, string data, 188–189

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 473

compile time, and polymorphism, 426–427
compilers
  defined, 4
  GNU C++, 11
  Japanese translation analogy, 164–165
compiler-supplied
  assignment operator function, 329–330
  constructors. *See* default constructors
  copy constructors, 305, 370–372
  subclass constructors, 395
compiling a C++ program, 9–10
composite data types, 364
compound statements
  containing single statements, 39, 76
  defined, 38
  in **do-while** statements, 229
  example, 73–76
  and initializing arrays, 137
  in **for** statements, 71–72
  in **while** statements, 69
computers, basic nature, 1–2, 37
concatenation
  String class example, 375–377
  string data, 167, 188–189
condition
  in **do-while** statements, 229
  in **if-else** statements, 3, 37
  in **for** statements, 66–67, 71
  in **while** statements, 44–45
console input. *See* **cin**
console output. *See* **cout**
const keyword, 304, 320
constants. *See* symbolic names, and **#define**
constructors
  in class declarations, 448
  copy constructors. *See* copy constructors
  default. *See* default constructors
  examples, 296–298, 299–301
  inlining, 292–293
  introduced, 250, 291–293
  multiple, 293
  naming, 292
  and subclassing, 395–397
  as virtual functions, 416
contained classes, example, 406–409
continue statements
  and **for / while** loops, 69, 444
  usage and syntax, 446
control structures, syntax summary, 443–445
  do-while, 444
  if-else, 443
  switch, 444
  while, 443
conversions between data types
  and binary files, 210
  in C, 36–37
  in FloatFraction subclass example, 399
  loss of data warnings, 34
  and operator functions, 328–329, 370, 381
  reinterpret_cast, 37
  and **sqrt** function, 210
  static_cast, 35, 88
  and subclasses, 397
convert to uppercase example, 177–179
converting temperatures example, 23–27
copy constructors
  and assignment operator functions, 329–330
  default, 370–372
  described, 304–306
  example, 306–310

copy constructors (*continued*)
  and references, 305–306
  and subclassing, 396
copying, deep / shallow, 370–372
cos function, 456
cosh function, 456
cout
  and class print function, 332–333
  as data object, 8, 15
  and file-stream objects, 198–199
  and polymorphism, 426–427
  string data, 189
creation, of objects. *See* constructors
current directory, 199–200

data
  class. *See* data members
  declaration, 446–447
  defined, 4–5
  storage of variables, 34, 163–164
data cast. *See* conversions between data types
data conversion functions, table, 454
data flow operators (<< and >>). *See* stream operators
data hiding, 250
data members
  assessing, 262
  in Fraction class, 270
  private. *See* private class data
  protected. *See* protected class data
  public. *See* public class data
  public / private, 262–263
  and **struct**, 263
data sharing, **extern** declarations, 232
data types. *See also* conversions between data types
  bool, 51, 331–332
  char, 165
  and classes, 249
  composite, 364
  discussion, 33–36
  double, 22
  file-stream objects, 198
  float, 23
  int, 35
  introduction, 21–23
  string. *See* string type
  struct. *See* **struct** types
  summary, 439–440
  table, 439–440
  void, 85, 104, 442
debugging, 10
decision-making, in programs, 37
declaration, data, 446–447
declaring, array of strings
  and **char*** notation, 123
  examples, 125, 183
declaring, arrays, 112
declaring, functions
  described, 82–86
  prototypes in #**include** files, 231–232
  usage and syntax, 447
declaring, pointers, 142–144
decrement operators, table, 52
deep copying, 370–372
default constructors
  assignment operator, 329–330
  introduced, 294–296
  String class example, 368
  and subclassing, 396–397
default statements (in **switch**), 230–231, 445
delete operator
  and clean-up, 365
  destroying arrays, 348
  introduced, 345
delimiters, 181
dereferencing pointers, 346–347
destroying allocated memory. *See* **delete** operator

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 475

destructors
  in class declarations, 448
  clean-up, 365
  introduced, 250, 364–366
  in String class example, 365–366
  variable scope, 365
  as virtual functions, 416
dice rolling, example, 106–109
directives, 16
directory, current, 199–200
disk files, 197–200. *See also* file-stream objects
display file from command line example, 221–223
display text file example, 203–205
dollars and cents, binary precision, 268
DOS shell, and RHIDE, 13
double precision floating point, 22
double-it function example, 145–147
do-while statements, 228–229, 444
dynamic allocation. *See* **new** operator
dynamic memory example, 349–350

efficiency
  and bounds checking, 136
  and inline functions, 320
  and pointers, 156, 345
  and references, 318–319
empty argument lists, functions, 85
empty function, of **string** class, 193
encapsulation
  introduced, 246–247
  and modules, 233–234
end of file (**eof**) function, 205
end user, defined, 6
endl, described, 17
eof function, 205
equality (==)
  class operator function, 330–331
  confusion with = (assignment), 38, 40–41
  String class example, 368–369
error propagation, 236–237, 241
errors, logic, 9–10
errors, runtime. *See* exception handling
escape sequences, 172, 200
even or odd example, 41–43
examples
  analyzing comma-delimited input, 181–185
  array elements, 115–116, 117–121
  array sorter, 149–153
  arrays of strings, 181–185
  binary files, 211–216
  Boolean values, 56–57
  building and running programs, 12–15
  building strings (with **string**), 189
  card dealer, 123–129, 130–135
  cin.getline, 175–177
  comma-delimited input, 181–185
  command line operations, 221–223
  compound statements, 73–76
  constructors, 296–298, 299–301
  contained members, 406–409
  convert to uppercase, 177–179
  converting temperatures, 23–27
  copy constructors, 306–310
  display file from command line, 221–223
  display text file, 203–205
  double-it function, 145–147
  dynamic memory, 349–350
  exception handling, 239–241
  FloatFraction class, 389–394, 398–399, 418–422
  floating point numbers, 23–27
  for loops, 70–71
  for statements, 68–69
  Fraction arithmetic: add and mult, 283–288
  Fraction class, 249–250

examples (*continued*)
Fraction class, 333–338
Fraction class constructors, 299–301
Fraction class copy constructor, 306–310
Fraction class operators, 323–328
Fraction class support functions, 276–279
FractionUnits class, contained members, 406–409
function usage, 88–95
get a number with **cin.getline**, 175–177
greatest common factor, 98–102, 239–241
if-else statements, 41–43
math library, 58–61
new keyword, 349–350
newlines, 18–19
odd or even, 41–43
operator functions, 320–328
overloading functions, 226–227
pass by reference, 144–147
Point class, 296–298, 320–323
pointers, 144–147, 149–153, 157–160
polymorphism, 418–422, 427–431
prime factorization, 102–106
prime number function, 92–95
prime number test, 58–61, 73–76
print 1 to N, 48–50, 70–71
Printable class, 427–431
printing a message, 12–15
printing array elements, 115–116
printing different type arrays, 226–227
printing multiple lines, 18–19
ProperFraction class, 399–403
random number generator, 106–109
random-access, 211–216
reading file-stream data, 203–205
reading from console, 175–177
recursive functions, 98–106
reference passing, 144–147
sorting arrays, 149–153
string building, 168–171
String class, 366–370, 377–381
string data, 189
StringParser class, 355–360, 427–431
strings, accessing individual characters, 177–179
strings and arrays, 123–129, 130–135, 181–185
swap function, 147–148, 152, 224–226, 303–304
temperature conversion, 23–27
testing a person's age, 56–57
testing randomness, 117–121
testing the Fraction class, 280–283
testing the Point class, 266–268
text file display, 203–205
triangle-number function, 88–92
virtual functions, 418–422
while loops, 48–50
writing to text file, 200–202
zeroing an array, 157–160
exception handling
centralized, 237
error propagation, 236–237
examples, 235, 239–241
exception types, 237–239
multiple try-catch blocks, 241
throw statements, 238
try-catch, 236–239
what function, 239
within functions, 235–236
exp function, 456
exponent field, in floating-point data, 34

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 477

expressions
 compared to statements, 53–54
 function calls as, 87
 syntax, 441–442
extern declarations, 232–233
ExtString subclass example, 386–387

fabs function, 456
Fahrenheit conversions, 23–27
false and true, 50–51, 55–56
fields, in binary files, 209
file handles. *See* file-stream objects, naming
file names, 200
files. *See* disk files; file-stream objects
file-stream objects. *See also* binary files
 closing, 199
 creating (opening), 198
 eof function, 205
 failure to open, 202
 fstream type, 198, 213
 ifstream type, 198
 input, output, generic, 198
 introduced, 197–199
 naming, 198
 ofstream type, 198
 opening, 198
 read and **write** functions, 207, 208–210
 sizeof function, 209
 text / binary mode, 198
 types, 198
 writing, 198–199
find function, of **string** class, 193
float data type, 23
FloatFraction subclass example
 adding constructors, 396–397
 completed code, 398–399
 declaration, 386
 initial code, 389–394
 and polymorphism, 413–415, 418–422
floating-point data
 example, 23–27
 introduced, 22
 storage, 34
floor function, 456
folder, current, 199–200
for statements
 compared to **while** statements, 69
 examples, 68–69, 70–71
 introduced, 66–67
 local variables, 72–73, 134
 usage and syntax, 444
Fraction class example
 adding and multiplying, 283–288
 completed example, 333–338
 constructors, 299–301
 conversion from integers, 328–329
 copy constructor, 306–310
 design, 249–250
 inline functions, 271–273
 introduced, 268–271
 operators, 323–328
 subclassing. *See* FloatFraction subclass example; Fraction Units subclass example; ProperFraction subclass example
 support functions example, 276–279
 virtual functions, 415–416
FractionUnits subclass example, 388, 406–409
friend functions
 and operator functions, 317–318
 ostream objects, 333
 Point class example, 317–318
 Printable class example, 430
fstream objects, 198, 213
function members. *See* member functions

function overloading
  array printing example, 226–227
  constructors, 293
  declarations, 225
  described, 224–226
  and object-oriented programming, 225–226
  operator functions, 315
  as polymorphism, 255
function usage examples, 88–95
functions. *See also* member functions
  arguments, 85, 86, 100–101
  behavior dictated by data type, 226
  calling, 83–84, 87–88
  calling themselves. *See* recursive functions
  categories, 82
  and complex software, 83
  concepts, 81–83
  declaring. *See* declaring, functions
  declaring vs. defining, 85
  defining, 84, 86
  execution flow (chart), 84
  friend. *See* friend functions
  inline, 271–273
  overloading. *See* function overloading
  parentheses required, 82
  pass by reference. *See* pass by reference
  prototypes. *See* functions, declaring
  recursive. *See* recursive functions
  using, 84–88
  virtual. *See* virtual functions

garbage collection, 348
GCF. *See* greatest common factor
generic file streams. *See* **fstream** objects
getline function
  in building strings example, 168–171
  interaction with >> stream operator, 174
  reading string data, 172–174
  string data, 189
  and text files, 207
global functions, and operator functions, 316–318
global variables. *See also* local variables
  anarchy in large programs, 246
  arrays initialized, 113, 165–166
  described, 95–96
GNU C++ compiler, 11
goto statements, 446
greatest common factor
  concepts, 273–275
  examples, 98–102, 239–241
  Fraction class example, 278
  introduced, 98–99
  zero divisor, 235

hexadecimal numbering system, 140–141
hierarchy, contained classes, 406
hierarchy, subclasses, 387
high-level languages, 6

IDE. *See* integrated development environments
if and **if-else** statements
  example, 41–43
  introduced, 37–39
  usage and syntax, 443
ifstream objects, 198
include files. *See* **#include**
includes, virtual, 16, 58
increment, in **for** statements, 66–67, 71

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 479

increment operators, 52. *See also* ++ (increment operator)
indexes, array
  introduced, 112
  nonexisting elements, 135–136
  zero-based, 114–115
indexing nonexisting array elements. *See* bounds checking
indirection operator (*)
  introduced, 142–144
  and passing variable references, 146–147
  and swap function, 148
inheritance. *See* subclassing
initializer, in **for** statements, 66–67, 71
initializing
  arrays, 113, 165–166
  functions. *See* constructors
  objects, 219–221
  string data, 187–188
  variables, 61
inline functions
  and efficiency, 320
  introduced, 271–273
  virtual, 416
input / output files. *See* file-stream objects
insert function, of **string** class, 193
instantiation, abstract classes, 424–425
insufficient memory, 348
integer data
  introduced, 22
  range, 35
  storage, 34
integrated development environments
  entering C++ programs, 8
  RHIDE, 11
  Visual Studio, 8
interfaces
  and abstract classes, 424–426
  introduced, 252–253
  and reusability, 258
ios mode flags, 213

Japanese translation analogy, 164–165
Java (computer language), 7

keywords
  in syntax diagrams, 3
  and variable names, 29–30

late binding, 256, 415
LCM. *See* lowest common multiple
leaks, memory, 348, 366
legal names, rules, 29–30
less than or equal to (=) operator, 49
lexical analyzer. *See* StringParser example
liberation, 433
linking a C++ program, 9–10
literals, syntax, 441
local variables, 134. *See also* global variables
  arrays uninitialized, 113, 165–166
  described, 95–96
  and function arguments, 100–101
  in function definitions, 86, 90
  in **for** statements, 72–73, 134
log, log10 functions, 456
logic errors, 9–10
looping, in programs
  break statement, 77
  continue statement, 69
  and counting, 65–66
  do-while loops, 228–229
  for loops, 66–67
  infinite loops, 48
  introduced, 44–47
  while loops, 44–47, 65–66
lower/upper-case letters. *See* case-sensitivity

lowest common denominator, 275–276, 284
lowest common multiple
  and adding fractions, 284
  concepts, 275–276
  Fraction class example, 278

machine code, defined, 5
magic boxes (variables), 20
main function
  command line arguments, 220, 223
  introduced, 14–15
  returning a value, 19
math library
  #**include**, 16
  example, 58–61
  functions, 455–457
  introduction, 57–58
member functions. *See also* constructors; functions
  accessing private data, 263–266
  calling, 173
  class declaration, 264
  Fraction class example, 270
  operator functions, 314
memory
  addresses, 140–142
  allocating. *See* **new** operator
  dynamic memory example, 349–350
  insufficient, 348
  leaks, 348, 366
  releasing. *See* **delete** operator
  segments, 142
Microsoft Visual Studio, 8, 13–14
mode flags, **fstream**, 213
models, real-world, 433
modules (program), 231–234
modulus function. *See* remainder (%) function
multiple modules, 231–234
multiplication-assignment operator (*=), 265. *See also* assignment operators

names, legal (in C++), 29–30
namespaces
  class, 250
  std, 17, 187, 190
new operator
  allocating arrays, 347–348
  allocating data, 343–345
  bounds checking, 364
  creating objects, 345–347
  dynamic memory example, 349–350
  example, 349–350
  insufficient memory, 348
  introduced, 343–345
  objects, 345–347
newlines
  and binary files, 206–207
  creating, 17
  example, 18–19
  \n (newline), 172
nonzero values, as true, 50
NOT Boolean operation (!), 54
null character (**\0**), 166, 168
null pointer, returning, 186
null value, 108, 201

object files (.o file), 334
object-oriented programming
  as C++ feature, 7–8
  communication by messages, 431–432
  introduced, 245–248
  overloading, 225–226
  real-world models, 433
  reusability, 257–258
  systems, 431–432
objects. *See also* classes
  argument list, 219–221, 249

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 481

cleanup, at object destruction, 250
constructors, 250
cout, 8
creation / destruction, 250. *See also* constructors; destructors
defined, 248–249
new operator, 345–347
pointers to, 345–347
resident knowledge, 254, 427, 432
odd or even example, 41–43
ofstream objects, 198
OOPS. *See* object-oriented programming
opening, files. *See* file-stream objects
operands, and operator functions, 314–315, 316
operator functions
<< stream operators, 332–333
efficiency and references, 318–320
examples, 320–328
as global functions, 316–318
introduced, 313–315
as member functions, 314–315
ostream objects, 332–333
operator overloading, 225–226, 255
operators (C++), 435–437
OR Boolean operation (||), 54
output files. *See* file-stream objects; **ofstream** objects
overloading. *See* function overloading; operator overloading
overriding
base class members, 386
in ProperFraction class, 402–403

parser. *See* StringParser example
Pascal (computer language), 6
pass by reference
example, 144–147
photocopy analogy, 146
and pointers, 139–140, 144
persistence, 414
placeholders, in syntax diagrams, 3
playing cards. *See* cards, dealing examples
Point class example
constructors, 296–298
efficiency and references, 318–319
friend functions, 317–318
introduced, 261–262
member functions, 264–265
operator functions, 314–315
operators, 318–319, 320–323
private data, 263–266
testing, 266–268
pointers
arithmetic operations, 154–155
and array processing, 156–157
comparing in String example, 368–369
concepts, 139–140
declaring, 142–144
dereferencing. *See* dereferencing pointers
examples, 144–147, 149–153, 157–160
and references, 302–304, 306
and string manipulation functions, 166
to unnamed variables, 344–345
using, 142–144
polymorphism
and **cout**, 426–427
defined, 413
examples, 413–415, 418–422, 427–431
introduced, 253–255
and reusable code, 432
and traditional languages, 256–257
and virtual functions, 255–257
void pointers, 426–427

pow function, 456
precedence
  of Boolean operators, 55
  C++ operators, 435–437
  and pointer arithmetic, 159–160
precision, storing dollars and cents in binary, 268
preprocessor directives, 16
prime factorization example, 102–106
prime number function example, 92–95
prime number test examples, 58–61, 73–76
Printable class example, 427–431
printing examples
  array elements, 115–116
  different type arrays, 226–227
  messages, 12–15
  multiple lines, 18–19
  1 to N, 48–50, 70–71
private class data. *See also* protected class data
  and class declarations, 448
  example, 263–266
  and **struct**, 263
  and subclasses, 393–394
private function members. *See* member functions
private keyword
  in class declarations, 448
  introduced, 263–264. *See also* private class data
program, defined, 1
programming, introduction, 1–6
programs, defined, 5
programs, logic errors. *See* logic errors
projects, in Visual Studio, 13
ProperFraction subclass example, 387, 399–403
protected class data
  and class declarations, 448
  introduced, 404–406
  and virtual functions, 414
protected keyword, 405, 448. *See also* protected class data
prototypes, function. *See* functions, declaring
pseudocode, described, 2–3
public class data
  base classes, 388–389
  and class declarations, 448–449
  public functions, 263–266
  and **struct**, 263
public keyword. *See also* public class data
  and C language, 296
  in class declarations, 448–449
  and **struct**, 289
pure virtual functions, 423–424

quotation marks, characters vs. strings, 180

rand function
  dice roll example, 107, 109
  usage and syntax, 457
random access, 211–216. *See also* binary files
random number generator example, 106–109
randomization functions, 457
rational number class. *See* examples, fraction class
read and **write** functions, 207, 208–210
reading file-stream data example, 203–205
reading from console example, 175–177
records, in binary files, 213

recursive functions
  described, 97–98
  examples, 98–106
  and inlining, 273
reference passing example, 144–147
references
  and assignment operator functions, 330
  and copy constructors, 305–306
  described, 302–304
  and indirection, 146–147
  and operator function efficiency, 318–320
  and pointers, 302–304, 306
  and returning new objects, 377
  swap function example, 303–304
reinterpret_cast, 37, 210
relational operators, 49, 50
releasing memory. *See* **delete** operator
remainder (%) function
  described, 42–43
  example for playing cards, 129
  using to find greatest common factor, 98–99
replace function, of **string** class, 193
reserving space in strings, 183
resource management issues, 363
return statements, 86
return values from functions, 81, 86
returning values from **main**, 19
reusable code, 257–258, 432
RHIDE development environment, 11, 12–13
runtime library. *See* C++ library
runtime memory allocation. *See* **new** operator

sample code, on CD, 11
scaling, array pointers, 155
scope, 365. *See also* global variables; local variables
scope prefix (::), 264–265, 292, 423
seed, random number, 108, 120, 456
seekp function, 216
segments, memory, 142
semicolons
  after aggregates, 113
  and class declarations, 262
  and compound statements, 38
  used to end C++ statements, 15, 53
shallow copying, 370–372
short-circuit logic, 54–55
sign bit, in data storage, 34
sin function, 456
single character functions, table, 455
single quotation mark, and strings vs. characters, 180
sinh function, 456
16-bit addresses, 141
size function, 192
sizeof function, 209, 210
sorting arrays example, 149–153
source code, defined, 5
sqrt function
  and conversions between data types, 210
  usage, 57–58
  usage and syntax, 456
srand function, 108, 120, 457
stack, function calls, 97–98
standard interfaces. *See* interfaces
standard library. *See* C++ library
statement blocks. *See* compound statements
statements
  compared to expressions, 53–54
  defined, 5
  multiple on a line, 53
  syntax, 442–443

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 484

static_cast, 35–36, 88
strcat function, 167–168, 453
strchr function, 358, 359, 453
strcmp function, 369, 453
strcpy function, 167–168, 453
strcspn function, 454
stream operators
  <<, and **cout**, 15
  >>, and **cin**, 20
  and **getline** function, 173–174
  operator functions, 332–333
  and text files, 207
streams (data). *See also* file-stream objects
  defined, 197–198
string building example, 168–171
String class example
  assignment operator functions, 374–375
  concatenation function, 375–377
  cpy function, 374–375
  default constructors, 368
  destructors, 368
  == (equality) function, 368–369
  final, 377–381
  full implementation, 377–381
  introduced, 363
  operator+ function, 375–377
  simple implementation, 366–370
  and **string** type, 363
  subclassing, 386–387
string data. *See also* **string** type
  accessing individual characters, 177–179, 191–192, 193
  as array of type **char**, 165–166
  and arrays, 122–123
  concatenation, 167, 188–189
  declaring as **char**, 123
  declaring as string, 187
  defined, 20
  delimited, 181
  example, 189
  initializing, 187–188
  introduced, 22
  library functions, 453–455
  local / global initialization, 165–166
  manipulation functions. *See* string manipulation functions
  and memory address (**string** type), 193
  memory layout, 163–164
  reading, 172–174
  reserving space in arrays, 183
  single/double quotation marks, 180
  terminator, 166, 168
string manipulation functions
  described, 166–168
  pointer arguments, 166
  string data, 188–189
  in string library (<string.h>), 166–168
  table, 453–454
string type. *See also* String class example
  described, 186–189
  example, 189–191
  member functions (table), 193
  and memory address, 193
  size function, 192
  and String class example, 363
StringParser example
  class code, 355–360
  constructors, 353
  design, 350–355
  member functions, 353
strings, accessing individual characters example, 177–179
strings and arrays examples, 123–129, 130–135, 181–185
strlen function
  compared to **size**, 192
  usage, 167
  usage and syntax, 454

strncat function, 167–168, 454
strncmp function, 454
strncpy function, 167–168, 454
strpbrk function, 454
strrchr function, 454
strspn function, 454
strstr function, 454
strtok function, 181, 248, 454
struct types
  and default constructors, 295–296
  and **public** keyword, 289
  syntax, 449
  treated as a class, 263, 296
structured exception handling. *See* exception handling
subclassing. *See also* inheritance
  adding more specialized abilities, 399–400
  and constructors, 395–397
  declaration, 448–449
  default constructors, 396–397
  hierarchy, 387
  introduced, 251–252, 385–388
  passing objects to base class, 427
  and public base class, 388–389
  and reusability, 258
  subclasses, 399
  type conflicts, 395, 397–398
subscripts, index. *See* indexes, array
swap function examples, 147–148, 152, 224–226, 303–304
swap function, of **string** class, 193
switch-case statements, 230–231, 444–445
symbolic names, and **#define**, 122
syntax diagrams
  assignment operator function, 330
  calling member functions, 173
  class, 261
  described, 3
  destructors, 364
  double data type, 22
  do-while statement, 229
  for statements, 67, 71
  function declaration, 85
  function definition, 86
  if statements, 37
  if-else statements, 39
  int data type, 35
  member function definition, 265
  new operator, 344, 346
  operator functions, 313
  pointer declaration, 142
  size function, 192
  static_cast, 36
  subclass declaration, 385
  summary, 441–449
  throw statements, 238
  try-catch. *See* exception handling
  two-dimensional arrays, 136
  while statements, 44
systems, and object-oriented programming, 433

tab (\t), 172
tan function, 456
tanh function, 456
temperature conversion example, 23–27
terminating null, 166, 168
testing a C++ program, logic errors, 10
testing randomness example, 117–121
text data. *See* string data
text editors
  and binary files, 208
  entering C++ programs, 8
  viewing text files, 201
text file display example, 203–205
text files, compared to binary. *See* binary files
32-bit addresses, 141–142

this keyword
  and assignment operator functions, 330
  described, 372–373
  and operator functions, 314–315
throw statements, 238
time function, and random seeds, 108, 457
tolower function, 179, 455
toupper function, 178, 179, 455
triangle-number function example, 88–92
true and false, 50–51, 55–56
try statements. *See* exception handling
try-catch exception handling. *See* exception handling
two-dimensional arrays. *See* arrays, two-dimensional
type conflicts, subclassing, 395, 397–398
type information, functions. *See* functions, declaring
types of variables. *See* data types

uninitialized variables, local and global arrays, 113
union keyword, 449
unnamed variables, 344
upper/lower-case letters. *See* case-sensitivity
user. *See* end user, defined
using statements, 16
  namespace std, 17, 187, 190

variables. *See also* global variables; local variables
  as buckets, 33
  declaring, 21
  external, 232–233
  initializing when declared, 61
  legal names, rules, 29–30
  as magic boxes, 20–21
virtual functions
  calling implementation, 416–418
  discussion, 415–416
  example, 418–422
  introduced, 413
  performance considerations, 416–418
  and polymorphism, 255–257
  and protected class data, 414
  pure, 423–424
  restrictions, 416
virtual includes, 16, 58
virtual keyword. *See* virtual functions
Visual Studio, 8, 13–14
void data type, 85, 104, 442
void pointers, and polymorphism, 426–427

warnings, loss of data, 34
what function, 239
while statements
  compared to for statements, 69
  example, 48–50
  introduced, 44–47
  usage and syntax, 443
white space, and **cin**, 173–174
write (and **read**) functions, binary files, 207, 208–210

zero-based indexing, 114–115
zeroing an array example, 157–160

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 488

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 489

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 490

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 491

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 492

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 493

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 494

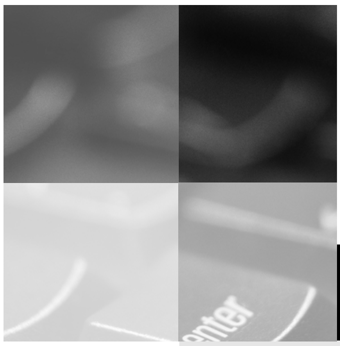
30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 495

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 496

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 497

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 498

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 499



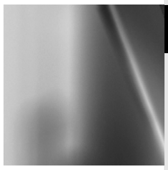
informIT

www.informit.com

YOUR GUIDE TO IT REFERENCE

Articles

Keep your edge with thousands of free articles, in-depth features, interviews, and IT reference recommendations – all written by experts you know and trust.



Online Books

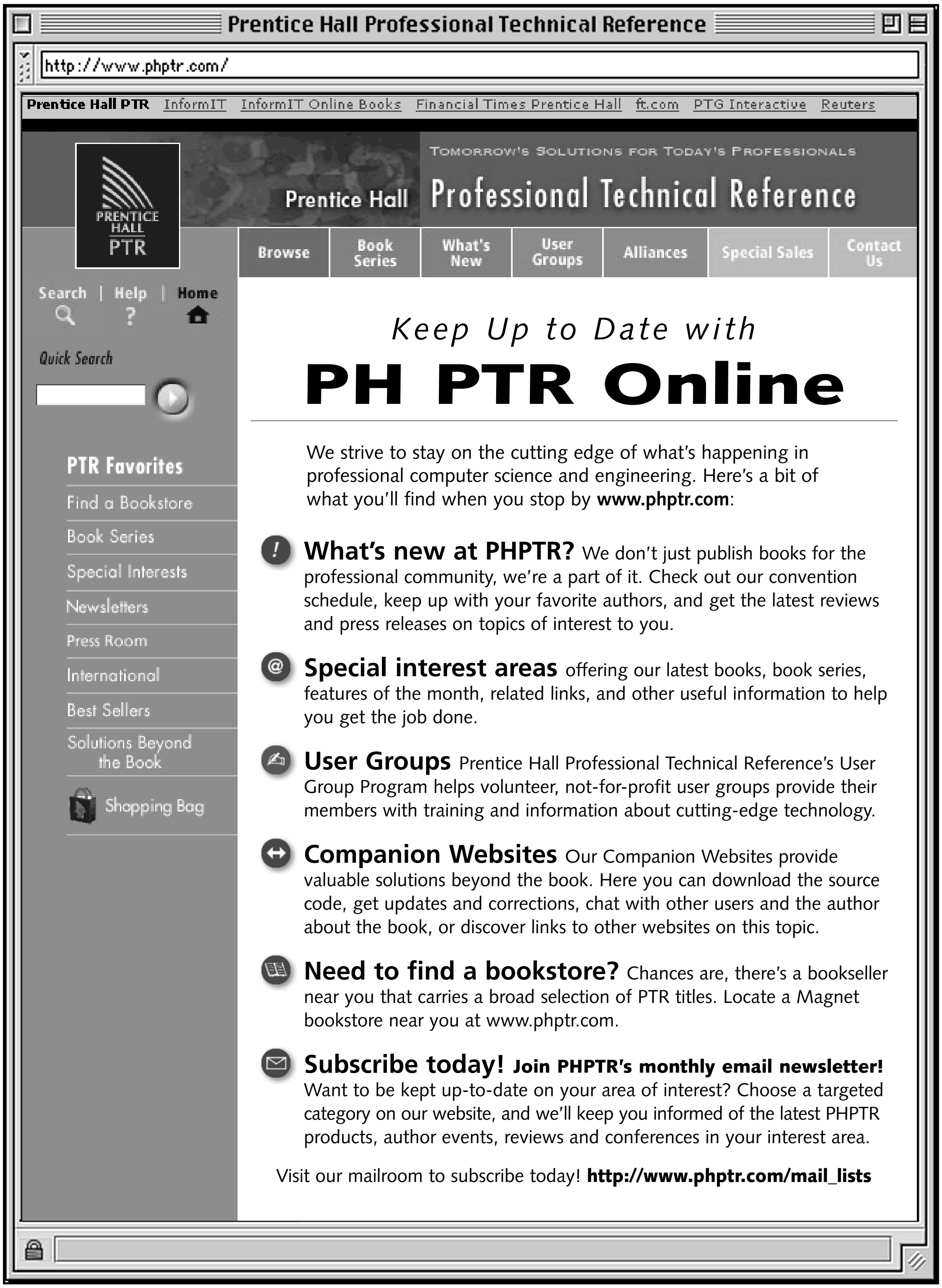
Answers in an instant from **InformIT Online Book's** 600+ fully searchable on line books. For a limited time, you can get your first 14 days **free**.

POWERED BY Safari® TECH BOOKS ONLINE®

Catalog

Review online sample chapters, author biographies and customer rankings and choose exactly the right book from a selection of over 5,000 titles.

30855 19 pp469-500 r2jm.ps 8/19/04 10:54 AM Page 500



Prentice Hall Professional Technical Reference
http://www.phptr.com/
Prentice Hall PTR InformIT InformIT Online Books Financial Times Prentice Hall ft.com PTG Interactive Reuters
Tomorrow's Solutions for Today's Professionals
Prentice Hall Professional Technical Reference
PRENTICE HALL PTR
Browse
Book Series
What's New
User Groups
Alliances
Special Sales
Contact Us
Search | Help | Home
Quick Search
PTR Favorites
Find a Bookstore
Book Series
Special Interests
Newsletters
Press Room
International
Best Sellers
Solutions Beyond the Book
Shopping Bag

Keep Up to Date with

PH PTR Online

We strive to stay on the cutting edge of what's happening in professional computer science and engineering. Here's a bit of what you'll find when you stop by **www.phptr.com**:

What's new at PHPTR? We don't just publish books for the professional community, we're a part of it. Check out our convention schedule, keep up with your favorite authors, and get the latest reviews and press releases on topics of interest to you.

Special interest areas offering our latest books, book series, features of the month, related links, and other useful information to help you get the job done.

User Groups Prentice Hall Professional Technical Reference's User Group Program helps volunteer, not-for-profit user groups provide their members with training and information about cutting-edge technology.

Companion Websites Our Companion Websites provide valuable solutions beyond the book. Here you can download the source code, get updates and corrections, chat with other users and the author about the book, or discover links to other websites on this topic.

Need to find a bookstore? Chances are, there's a bookseller near you that carries a broad selection of PTR titles. Locate a Magnet bookstore near you at www.phptr.com.

Subscribe today! Join PHPTR's monthly email newsletter! Want to be kept up-to-date on your area of interest? Choose a targeted category on our website, and we'll keep you informed of the latest PHPTR products, author events, reviews and conferences in your interest area.

Visit our mailroom to subscribe today! **http://www.phptr.com/mail_lists**