Index

Symbols and Numerics

&& (AND) operators, 76, 77, 80
(–>) arrow operator, 255, 398
= (assignment) operator, 31, 57–58, 73
\\ (backslash) character, 45
/* characters, 28
–– (decrement) operators, 56, 88–89
. (dot) operator, 398
// (double slash) comment characters, 27
== (equality) operator, 72, 73
<< extraction operator, 399
(>) greater-than operator, 73, 77
(>=) greater-than-or-equal-to operator, 73
++ (increment) unary operators, 56, 88–89
>> insertion operator, 399
(<) less-than operators, 73
(<=) less-than-or-equal-to operator, 73
– (minus) unary operator, 55
% (modulus) binary operator, 398
~ (NOT) bitwise operators, 80, 294
|| (OR) operators, 76, 77, 80, 81
++ (plus-plus) unary operator, 56
+ (plus) unary operator, 55
:: (scope resolution) operator, 225–226, 398
symbol, 184
?: (ternary) operator, 398
* (unary) pointer operator, 150–152, 254–255
& (unary) pointer operator, 150–152

A

abstract classes, 431
  concept of, 388–390
  passing, 391–392
  subclasses of, 391
accumulator value, 95, 98
addCourse(), 221–226, 283
addition (+) binary operator, 54
add() method, 357
AND (&) bitwise operators, 80
AND (&&) logical operators, 76, 77
arguments, 111, 116
  default values, 107, 117–118
  multiple arguments, 111
  nDiameter. argument, 106
  passing by reference, 118–120
  passing by value, 118–120
ArrayDemo sample program, 125–127
  displayArray() function, 127, 128
  readArray() function, 127–129
  sumArray() function, 127, 128
ArrayOfStudents sample program, 251–252

COPYRIGHTED MATERIAL

arrays
accessing individual elements of, 124
buffer overflow, 144
avoidance of, 144–145
of characters, 134–135
concatenating strings, 137–139
creating string of characters, 135–137
strings and, 236
of character strings, 177–179
constructing, 321–322
declarations, 124, 129
definition of, 123, 124
deleting, 298
error handling, 129
example of, 124–128
initializing, 129–130
integerArray, 130
library functions, 141–142
matrix configuration, 139–141
nValueArray[i], 125
of objects
allocating off heap, 262–263
constructing, 291
declaring, 251–253
of pointers, 176–177
range-based for loop, 131, 133
relationship to pointers, 168–170
simple int vs array of ints, 125
size of, 130
span, 132–134
string-handling functions, 142
wide string-handling functions, 146–147
arrow (–>) operator, 255, 398
artificial intelligence (AI), 469
assert() function, 195, 196
assignGrades(), 285
assignment operator (=), 31, 57–58, 73
assignment operators
copy constructors vs, 399–400
deleting, 405–406
move constructor and, 407–410
overloading, 401–405
return type and value, 404
auto keyword, 49
automatic copy constructor, 331–332
automatic declarations, 49
automatic variables, 129

B

backslash character (\\), 45, 415
binary operators, 52–53
bitwise operators, 76, 79
NOT (~) operators, 80
AND operators (&), 80
OR operators (|), 80, 81
single-bit operators, 80–81
XOR operators, 81
bool data type, 40, 42
Boole, George, 73
BoolTest program, 73–74
bool (Boolean) variables, 73–74
BranchDemo sample program, 84–85
branch statements, 84–86
break command, looping statement, 94–96
BreakDemo program, 94–95

`break` statements, 100
`Budget` sample programs, 391–392
buffer overflow, 144
  avoidance of, 144–145
bug avoidance, 457
  avoiding multiple inheritance, 464
  avoiding overloading operators, 462
  clear and consistent coding style, 458–459
  commenting, 461
  enabling all warning and error messages, 458
  limiting visibility of class internals, 459–461
  managing heap memory systematically, 462
  single-stepping programs, 461–462
  using exceptions to handle errors, 463
  virtual destructors, 463–464

C

C++, 246, 276, 283, 413
  advanced features, 471
  basic concepts, 10–11
  character types, 46
  classes in, 397
  common escape sequences, 63, 64
  composition, 431–432
  creation
    building program, 23–24
    cheating, 22
    entering code, 20–21
    project creation, 18–19
  C-style string/C string, 136, 143
  exception handling, 436, 438, 441
  exception mechanism in, 463
  execution, 24–25
  expression, 30–32
  inheritance, 432
  interface, 431
  reviewing
    basic framework, 25–27
    clarifying source code, 27–28
    C++ statements, 28
    generating output, 29–30
    writing declarations, 29
  short-circuit evaluation, 77–78
  span, 132–134
  special characters, 44–45, 63, 64
  string, 235
  variable types, 39–40
  zero-terminated string, 137
C++17 standard, 245, 352, 446
C++2011 standard, 45, 49, 327
  constructing members with initializers, 315–316
  copies of objects, 328–329
  default constructor, 308–310
  delegating constructors, 307–308
  deleting assignment operator, 405–406
  deleting copy constructor, 340, 405–406
  `explicit` keyword, 323
  `final` keyword, 377–378
  invoking constructors other than default, 321–322
  library, 273

C++2011 standard *(continued)*
move constructor, 327, 340–341, 407–410
`nullptr` constant, 265, 404
`override` descriptor, 377–378
reflection, 380–382
return optimizations, 341
static member functions, 355
C++2020 standard, 78, 96, 246
concept of ranges, 96
modules, 201, 204
span, 132–134
C++2023 standard, 45, 47, 191, 192, 194, 449
modules, 201, 204
C++2026 standard, 11, 15, 32, 47, 250, 339, 344, 380, 382
concepts, 205
contracts in, 452–454
modules, 201
span, 132–134
`calculateAge()` function, 451–452
calculation speed, floating-point variables, 38
`CallMemberFunction` sample program, 220–223
`CallStaticMember` sample program, 353–355
`CascadingException` sample program, 439–441
`catch` keyword, 436–437, 439, 441–442
character array, 236–238, 249
character-based functions, 245
character types, 46
`CharDisplay` sample program, 134–135
char string, 123
`char` (character) variables, 40, 41, 44–46
`Checking` classes, 384–387
`cin` operator, 63
classes, 232–234
abstract
concept of, 388–390
passing, 391–392
subclasses of, 391
accessing members of, 217–218
defined, 216, 287–288
member functions
accessing other members from, 223–224
calling, 221–223
defining, 217–218, 220–221
defining in classes, 226–228
defining separately from classes, 229–230
identifying current object, 224–225
overloading, 230–231
naming conventions, 216
objects vs, 287–288
private members
function of, 277–278
general discussion, 275–276
getting friendly with your, 283–286
need for, 276–277
protecting internal state of classes, 279–280
using classes with limited interfaces, 280
scope resolution, 225–226

`class` keyword, 216
creating objects from, 216
format of, 216
class members, 348. *See also* static members
Code::Blocks environment, 10, 11, 26, 181, 459
64-bit programs, 11, 25
building program, 23–24
coding errors, 23
creating project, 18–19
installation
for Macintosh, 17
for Microsoft Windows, 12–14
for Ubuntu Linux, 15–16
Management window, 20
coding, 465
AI to write, 469
already written, 468
avoid fixing, later, 469–470
avoid hard-coded values, 471–472
comments use in, 466–467
formatting, 459
modern features in, 471
picking wise names for, 466
style, 458–459, 469
testing, 470
third-party, 469
use version control, 472
writing modular, 468
comments, 27–28
creating while coding, 461
use in code, 466–467
common escape sequences, 63, 64
comparison operators, 77
assignment operator (=), 73
`bool` variable, 73–74
equality operator (==), 72, 73
greater-than (>) operator, 73, 77
greater-than-or-equal-to operator (>=), 73
less-than (<) operators, 73
less-than-or-equal-to operator (<=), 73
logical `int` variables, 74
three-way comparison operator, 78–79
compiler, definition of, 10
compiler flag, 13
compile-time type, 374
compiling, definition of, 10
`Complex operator*(const Complex&, const Complex&)`, 398
composition, C++ development, 431–432
computer language (machine language), 9
`concatCharString()` method, 236–238
`Concatenate` sample program, 137–139
wide-character version of, 146–147
concrete classes, 388–389
concrete functions, 389, 390
conditional clause, for loop, 90
conditional compilation, 192
constant expressions, 99
constant pointer, 156–158
constant types of, 42–43
constant value, definition of, 42
`constexpr` function, 454
`const` keyword, 156–158
`const` reference, 339

ConstructArray sample program, 291
ConstructDataMembers sample program, 314
ConstructingMembers sample program, 311–312
constructing subclasses, 367–368
ConstructMembers sample program, 292–294
ConstructMembersWithInitializers sample program, 315–316
constructors
  cannot be virtual, 378
  combining, 307–308
  constructing arrays, 321–322
  constructing complex data members, 310–316
  constructing constant data members, 316
  constructing duplexes, 292–294
  constructing members with initializers, 315–316
  constructing multiple objects, 291
  constructing single objects, 289–291
  copy, 327
    automatic, 331–332
    example of, 328–331
    reasons for using, 329
    returning the object, 343
    shallow vs deep copies, 332–337
    temporary objects, 337–339
  default, 308–310, 312
  defined, 289
  defining with arguments, 302–304
  delegating, 307–308
  ignoring the return value, 325–326
  inheritance of by subclasses, 369
  move, 340–343, 407–410
  multiple inheritance, 430–431
  overloading, 304–308
  reverse order, 321
  rules for order of construction
    global objects, 317–321
    local objects, 317
    order of declaration, 320–321
    overview, 317–321
    static objects, 317–319
  type conversion, 322–324
ConstructorsCallingEachOther sample program, 307–308
ConstructorWArg sample program, 302
ConstructorWDefaults sample program, 306–307
ConstructSeparateID sample program, 312–313
ConstructStatic sample program, 317–319
const value, 352
const variables, 42
continue command, looping statement, 96
ContinueDemo program, 96
contract_assert(), 452–453
contracts
  conditions, 453
  defined, 452–453
Conversion project
  creation
    building program, 23–24
    cheating, 22

entering code, 20–21
project creation, 18–19
execution, 24–25
expression, 30–32
reviewing, 25–30
copy assignment operator, 399–400, 411
copy constructors, 327, 400, 404–408, 411
automatic, 331–332
example of, 328–331
reasons for using, 329
returning the object, 343
shallow vs deep copies, 332–337
temporary objects, 337–339
CopyConstructor sample program, 329–330
copyDArray() method, 404
counter () function, 346
counting, floating-point variables, 38
count variable, 346, 347
Course constructor, 293
cout object, 60–62, 64, 69
create() function, 462
C++ Standard Library, 413, 468–469
C++ statements, 28
C-style string/C string, 136, 143
cube() functions, 203
CustomExceptionClass sample program, 442–444

D

DArray class, 403, 406
data(int) function, 460
data members, 345
getting and setting, 280–283
static data members, 347
declarations, 29
arrays, 124, 129
automatic, 49
prototype, 115, 116
declaring variables, 160
constants, 42–43
int variable type, 35–36
overview, 34
types, 39–45
decltype() keyword, 49
decrement (––) unary operators, 56, 88–89
post-decrement version, 89
pre-decrement version, 89
DeepCopy sample program, 335–337
DefaultCopyConstructor sample program, 331–332
default keyword, 309–310
#define commands
advantage, 185
const variable, 189
definition, 184
enumeration, 189–192
inline function, 188
macro definitions, 186–188
nbr++ operations, 187
deleteDArray() method, 404
delete keyword, 263, 295, 298–299, 310, 334, 340
DemoAssignmentOperator sample program, 401–405
DemoMoveOperator, 410
demotion, 47

deposit() function, 219, 220, 229, 230
destruction, 298–299
DestructMembers sample program, 295–298
destructors, 411
  destructing subclasses, 368–369
  reasons for using, 294
  using, 294–298
  virtual, 379–380, 463–464
determinePizzaArea() function, 106, 107, 110, 111, 114
directory_iterator() method, 416
displayArray() function, 127, 128, 131, 135, 170–172
displayCharArray() function, 135
displayInstructions() function, 110
DisplayString sample program, 136, 172–173
dLocalVariable function, 160
Dog, 288
dot (.) operator, 398
double slash (//) comment characters, 27
double (double precision) variables, 37, 47
do...while loop, 88
drinkBeer(), 277
dynamic type, 374
dyn() function, 356

E

early binding, 374
editor, definition of, 10
elementsInName() method, 355
#elifdef commands, 193, 194
#else command, 192
#endif command, 192
endl value, 62
equality (==) logical operator, 72, 73
erase(), 243
error(), 451
#error command, 195
error messages, 23, 24, 131, 195, 438, 444, 458
error reporting process, 23–24
error returns, 438
exception class, 446
exception handling mechanism, 463
  examining, 439–442
  example, 436–437
  reasons for using, 438
  rethrowing exceptions up the chain, 445–446
  throwing objects, 442–444
exceptions
  defined, 436
  handling of error conditions, 436
executable files (.exe), 10
exists() method, 417
expected template, 449–451
expected value, 451–452
explicit keyword, 323
expressions
  mathematical operations, 54
  mixed-mode expressions, 47, 48
extended function names, 112–114
extraction operator (>>), 62, 63, 399

F

FactorialException sample program, 436–437
factorial() function, 436–438
factoring
  abstract classes
    concept of, 388–390
    passing, 391–392
    subclasses of, 391
  culling common properties from similar classes, 386–387
  defined, 383
  direct inheritance vs, 384–387
  tuple, 392–393
_FILE__ constant, 444
fileFunc(), 445
filename(), 416
filePath, 416–417
files, 413–414
  existence, 417
  filesystem, 414, 416, 417, 419
  metadata, 414, 418–419
  path, 414–416
filesystem, 414, 416, 417, 419
final keyword, 377–378
find() method, 243
FloatAverage sample program, 38, 39
floating-point constants, 37, 42, 47
floating-point (float) variables
  definition of, 37
  limits of
    calculation speed, 38
    counting, 38
    loss of accuracy, 38–39
    range, 39
  logical operations, 75
flow control statements
  branch statements, 84–86
  looping statement
    autoincrement/autodecrement feature, 88–89
    break command, 94–96
    continue command, 96
    for each loop/range-based for loop, 93
    infinite loop, avoiding, 92–93
    for loop, 89–92
    nested loop, 97–98
    overview, 86
    while loops, 86–88
  switch statement, 98–100
fn() function, 120, 121, 153–155, 297, 331, 334, 338, 349
fn1() function, 342–343
fn2() function, 339, 342–343
ForDemo1 program, 90–91
for each loop, 93
for loops
  conditional clause, 90
  example of, 90–92
  format of, 89
  initialization clause, 90
  problems, 91
  range-based for loop, 131
format() function, 246–248

formatting, 246
adding modifiers, 247–248
codes, 66–68, 247
forward declarations, 284–285
forward slashes, 415
`friend` keyword, 283–286
`__func__` constant, 444
functional programming
abstraction, 210–211
classification, 212–213
member functions, 219–220
function declaration, 117, 188
functionName function, 106
functions
with arguments, 111, 116
default values, 107, 117–118
multiple arguments, 111
`nDiameter` argument, 106
passing by reference, 118–120
passing by value, 118–120
benefits of, 121–122
data value and, 345–347
definition of, 103
`determinePizzaArea()`, 110
`displayInstructions()`, 110
example of, 104–110
exploring, 105–110
general form, 106
`getPizzaSize()`, 110
`main()`, 112, 115, 116
operators vs, 398–399
overloading names, 112–114
passing objects to
calling functions with object pointers, 196–197
calling functions with object values, 256–257
calling functions with reference operator, 258–259
memory consumption and copying objects, 260
prototypes, 115–116
returnType, 106
`showTheDetails()`, 110
simple function, 107
visible variables, 120–121
`void`, 106, 107

G

gcc compiler, 10
installation, 15
range of numeric types, 43–44
version 14.2.0, 11
`get` function, 280
`getline()` function, 144, 145
`getName()` method, 354, 356
`getPizzaSize()` function, 110, 121
`getPop()` function, 192
`GetSetStudent` program, 281–282
`getStudentCount()` function, 353, 354
getter functions, 280–282
Git, 472
global objects
defined, 288, 317
going out of scope, 295
rules for order of construction, 317–321
`globalVariable` variable, 120

GNU Project, 10
grade(double), 280
grades() function, 280
GraduateStudent sample program, 364–365
greater-than (>) operator, 73, 77
greater-than-or-equal-to operator (>=), 73

H

hard-coded numbers, avoid using, 471–472
HAS_A relationships, 369–370, 432
has_value() function, 448–449, 451–452
headers vs modules, 204
head pointers, 265–266, 351, 357
heap memory, 161, 164, 165
  allocating arrays of objects off, 262–263
  allocating objects off, 261–263
  assignment operator, 399
  classes that automatically allocate, 263–264
  constructors and destructors, 290, 294–298, 333–335
  exception handling mechanism, 445–446
  managing systematically, 462

I

#if commands, 192–194
  definition of, 184
#ifdef commands, 193
#ifndef command, 193
if statement, 84, 85, 100
#include command, 185, 197–201
  include files, 228
#include <cstring> statement, 142
increment (++) unary operators, 56, 88–89
indentation, 20
infinite loops, 92–93
inheritance, 372
  C++ development, 431–432
  defined, 361–362
  example, 364–365
  factoring
    abstract classes, 388–392
    culling common properties from similar classes, 386–387
    defined, 383
    direct inheritance vs, 384–387
    tuple, 392–393
  HAS_A relationships, 369–370, 432
  IS_A relationships, 362–363, 383, 432
  multiple inheritance mechanism
    avoiding, 464
    disadvantages of, 432–433
    name collision, 423–424
    object construction, 430–431
    overview, 421–423
    virtual inheritance, 424–430
  need for, 362–363
  polymorphism
    constructors, 378
    declaring functions as not overrideable, 377
    declaring functions virtual, 375–376
    defined, 374
    destructors, 379–380
    need for, 374–375

inheritance *(continued)*
overriding base class functions, 371–374, 376–378
`static` member functions, 378
subclasses, 366
constructing, 367–368
destructing, 368–369
inheriting constructors, 369
transitive nature of, 363
`InheritanceExample` sample program, 364–365
`init()`, 289
initialization clause, for loop, 90
`initialize()`, 283
initializer lists
arrays, 130
defined, 321
invoking constructors other than default, 321–322
inline functions
advantages of, 228
defining member functions in classes, 226–228
`inline` keyword, 352
`insert()` function, 243
insertion operator (`>>`), 60, 61, 63, 66, 399
`insertPhrase()` method, 243
instance members, 348
instances, defined, 216
instantiate, 390–391
`integerArray`, 130
integers, 53
definition of, 35
integer function, 107
limited range, 36
round-off, 35–36
signed version, 40
span, 133–134
unsigned version, 40
integrated development environments (IDEs), 69, 70, 184
interface
C++, 431
defined, 431
`int operator+(int, int)`, 398
`intReader` variable, 150
intrinsically defined objects, 194–196
intrinsic constants, 194
intrinsic data types, 397
intrinsic variables, 251
`int` variable type, 48
limitations of, 37
limited range, 36
logical operations, 74
truncation, 35–36
IS_A relationships, 362–363, 383, 432
`isLegal` sample program, 117
`.ixx` extension, 203

J

JetBrains CLion tool, 17

L

late binding. *See* polymorphism
`Layout` sample program, 151
`lcd()` function, 463
less-than (`<`) operators, 73

less-than-or-equal-to operator (<=), 73
level of abstraction, 210
limited visibility, 459–461
__LINE__ constant, 444
LinkedListClass sample program, 356
LinkedListData sample program, 267–270
linked lists
  adding objects to head of, 266
  advantages of, 264
  declaring linkable classes, 266
  defined, 264
  example of, 266–270
  head pointers, 265
  moving through elements in, 266–267
linking, definition of, 10
lists
  defined, 321–322
  initializer
    arrays, 130
    invoking constructors other than default, 321–322
  linked
    adding objects to head of, 266
    advantages of, 264
    declaring linkable classes, 266
    defined, 264
    example of, 266–270
    head pointers, 265
    moving through elements in, 266–267
local objects
  defined, 288
  going out of scope, 295
  rules for order of construction, 317
localVariable variable, 120
logical operations
  comparison operators, 77
    assignment operator (=), 73
    bool variable, 73–74
    equality operator (==), 72, 73
    greater-than (>) operator, 73, 77
    greater-than-or-equal-to operator (>=), 73
    less-than (<) operators, 73
    less-than-or-equal-to operator (<=), 73
    logical int variables, 74
    three-way comparison operator, 78–79
  on floating-point variables, 75
  logical operators
    bitwise operators, 76, 79–81
    AND operators (&&), 76, 77, 80
    OR operators (||), 76, 77, 80, 81
    short-circuit evaluation, 77–78
logical operators
  bitwise operators, 76, 79–81
  AND operators (&&), 76, 77, 80
  OR operators (||), 76, 77, 80, 81
  short-circuit evaluation, 77–78
logMessage() function, 192–193
long int (long integer) variables, 43
looping statement
  autoincrement/autodecrement feature, 88–89
  break command, 94–96
  continue command, 96
  for each loop/range-based for loop, 93

looping statement *(continued)*
infinite loop, avoiding, 92–93
`for` loop, 89–92
nested loop, 97–98
overview, 86
`while` loops, 86–88

M

machine-executable files, 10
machine language (computer language), 9
Macintosh
development tool installation, 17
Xcode installation, 17
Macintosh OS X, 11
`main()` function, 112, 152–154, 159, 171, 179–180, 238, 243, 282, 283, 297, 330–331, 343, 366, 437, 441, 452, 453
rules for order of construction, 319
`makeACopy()` function, 376–377
`make_shared()` function, 162
mathematical operations, 31, 51, 61
applying to pointer variables, 167–168
assignment operator (=), 31, 57–58
binary operators, 52–53
expressions, 54
order of precedence, 52–55
unary operators, 52, 55–57
matrix configuration of arrays, 139–141
`max_size()` function, 241
member functions, 345, 384–385, 389–390
accessing other members from, 223–224
calling, 221–223
constructors
cannot be virtual, 378
combining, 307–308
constructing arrays, 321–322
constructing complex data members, 310–316
constructing constant data members, 316
constructing duplexes, 292–294
constructing members with initializers, 315–316
constructing multiple objects, 291
constructing single objects, 289–291
copy, 327–340
default, 308–310, 312
defining with arguments, 302–304
delegating, 307–308
move, 328, 340–343, 407–410
multiple inheritance, 430–431
overloading, 304–308
rules for order of construction, 317–321
type conversion, 322–324
defined, 217–221, 289
defining in classes, 226–228
defining member functions in classes, 226–228
defining separately from classes, 229–230
destructors
reasons for using, 294
using, 294–298
virtual, 379–380
functional programming, 219–220
identifying current object, 224–225
overloading, 230–231
overriding base class functions, 371–374

static, 352–355
static, 378
virtual
constructors, 378
declaring, 375–376
declaring functions as not overrideable, 377
destructors, 379–380
overriding base class functions, 376–378
pure, 389–390
metadata, 418–419
methods, 221. *See also* member functions
Microsoft Windows, Code::Blocks installation, 12–14
microwave, 275–276
MinGW Compiler Suite, 12
minus-minus unary operator (--), 56
minus unary operator (-), 55
mixed-mode expressions, 47, 48
modern features, in code, 471
modifiers, adding, formatting, 247–248
modular code, 468
modules
benefits of, 202
in C++20 standard, 201, 204
in C++23 standard, 201, 204
in C++26 standard, 201
compiling, 205
creation of, 202–203
definition of, 201
vs headers, 204, 458
.ixx extension, 203
to use, 204
modulus (%) binary operator, 53
syntax changes, 398
move assignment operator, 407–411
move constructors (X::X(X&&)), 328, 340–343, 407–411
Move sample program, 341–342
MultipleInheritanceFactoring sample program, 425–430
multiple inheritance mechanism, 421
avoiding, 464
disadvantages of, 432–433
name collision, 423–424
object construction, 430–431
overview, 421–423
virtual inheritance, 424–430
MultipleInheritance sample program, 421–423
mySpan, 132

N

name collision, 199, 201, 423–424
NameDataSet sample program, 356–357
namespace, 199–201
naming conventions, 48
nbrOfStudents, 348–349, 351, 352
Neighborhood Memory model, 168
NestedDemo program, 97–98
nested loop, 97–98
new keyword, 163, 165
newline character (\n), 45, 69
nextStudent() function, 351
\n newline character, 45, 69
nonmember functions, 221

nonprintable characters, 44, 45
nonstatic member, 354, 356
nonvirtual base classes, 431
NOT (~) bitwise operator, 80, 294
notMe, 288
now() function, 451
npos constant, 243
null character (\∅), 136, 142, 172
nullptr constant, 176, 265, 404, 410
null-terminated byte string (NTBS), 136, 137, 172
numbers to strings, 249–250
number systems, 2
numeric types, range of, 43–44

O

object members, 348
object-oriented (OO) languages, 362
object-oriented programming (OOP), 275–277
  abstraction, 210–211
  classification, 211–212
  encapsulation, 214
  inheritance, 214
  polymorphism, 214
objects
  allocating off heap memory, 263–264
  arrays of
    allocating off heap memory, 262–263
    declaring, 251–253
  classes vs, 287–288
  creating from classes, 216
  defined, 287–288
  global
    defined, 317
    going out of scope, 295
    rules for order of construction, 317–321
  local
    going out of scope, 295
    rules for order of construction, 317
  naming conventions, 216
  object pointers
    arrow pointers, 255
    declaring, 253–254
    dereferencing, 254–255
  passing objects to
    calling functions with object pointers, 196–197
    calling functions with object values, 256–257
    calling functions with reference operator, 258–259
  static
    rules for order of construction, 317–319
ObjPtr sample program, 253–254
.o extension, 184
OOP. *See* object-oriented programming
operator overloading, 397–398
operators
  functions vs, 398–399
  insertion and extraction, 399
  overloading, 398–399, 462
  shallow copies, 399–400
optional template, 446–449
order of precedence, 52–55
OR operators (||), 76, 77, 80, 81
otherFunction(), 445

outline functions, 229–230
OverloadConstructor sample program, 304–305
overloaded operators
in string class, 241
overloading, 371–372
assignment operators, 401–405
constructors, 304–308
member functions, 230–231
operators, 398–399, 462
subscript operator, 406–407
OverloadOverride sample program, 372–373, 375
override descriptor, 377
overriding base class function, 371–374

P

padding, 67
parent_path(), 416
pArray, 404, 407
PassObjPtr sample program, 257–258
PassObjRef sample program, 258–259
PassObjVal sample program, 256–257
pHead pointer, 357
pickOne(), 288
PizzaSizer program, 104–111, 114, 121
plus-plus unary operator (++), 56
plus unary operator (+), 55
pMyString pointer, 161, 162
pNext pointer, 357
pointer-related operators, 150–151
pointer variables
adding offsets, 169
addresses, 150, 151
applying operators to, 167–168, 174
and arrays, 168–170, 174–175
applying operators to addresses, 170–172
constant pointer, 156–158
definition of, 149, 151
dereferencing, 173
head pointers, 265
incrementing, 173
link pointers, initializing, 265, 269
memory allocation, 160–161, 164–166
object pointers
arrow pointers, 255
declaring, 253–254
dereferencing, 254–255
operators, 150–152
passing arguments to functions, 154
by reference, 155–156
by value, 153–154
returning from function, 158–160
smart pointers, 161–164
string manipulation, 172–173
polymorphism (late binding)
constructors, 378
declaring functions as not overrideable, 377
declaring functions virtual, 375–376
defined, 374
destructors, 379–380
need for, 374–375
overriding base class functions, 371–374, 376–378
static member functions, 378

post(), 453
power() functions, 203
pre(), 453–454
preprocessor
  compile-time feature, 205
  #define commands, 184–192
  executable programs, 184
  #if commands, 192–194
  #include commands, 197–199
  intrinsic constants, 194–196
  linker, 184
  modules, 201–205
  overview of, 184
  typedef keyword, 196–197
  using keyword, 196–197
PrintArgs program, 180–181
print() functions, 64–66, 69, 103, 106, 247–248, 471
printing
  formatting codes, 66–68
  pros and cons, 69
  simple printing, 64–65
  special characters, 63, 64
  traditional method of, 60–64
  variables, 65–66
println() function, 64–66, 68, 69, 106, 246–248, 282, 382
println(line), 249
private class, 276
private keyword, 276, 277, 366, 459
private members
  function of, 277–278
  general discussion, 275–276
  getting and setting data members, 280–283
  need for, 276–277
  protecting internal state of classes, 279–280
  using classes with limited interfaces, 280
programs, ways to understand, 465
  AI to create code, 469
  avoid fix the code later, 469–470
  avoid hard-coded values, 471–472
  comments, use of, 466–467
  modern features, 471
  picking wise names, 466
  styling your code, 469
  testing, 470
  version control, 472
  writing modular code, 468
promotion, 47
protected class, 276
protected keyword, 276, 278, 366–367
protected members
  declaring constructors as, 340
  limiting visibility of class internals, 459
prototype declaration, 115, 116
public class, 276
public keyword, 216, 277, 366

R

range-based for loop, 93, 131, 133, 174
range logic, 416–417
readArray() function, 127–129
real-world accounts, 218–219
Red Hat Linux, 15

reflection features, 380–382
registration() function, 284
release() function, 462
removeSpaces() method, 243
replace_extension() method, 416
resolving calls, 231
returnType indicator, 106
return value optimization (RVO), 343–344
reverse order
  constructors, 321
round-off error, 75
rule of five, 411
rule of zero, 411
runtime type, 374
RVO. *See* return value optimization (RVO)

S

SavingsAccount, 216, 217, 219
Savings classes, 384–387
SavingsClassOutline sample program, 229–230
scope resolution (::) operator, 225–226, 398
security
  exception handling mechanism
    examining, 439–442
    example of, 436–437
    reasons for using, 438
    rethrowing exceptions up the chain, 445–446
    throwing objects, 442–444
  multiple inheritance mechanism, 421
    avoiding, 464
    disadvantages of, 432–433
    name collision, 423–424
    object construction, 430–431
    overview, 421–423
    virtual inheritance, 424–430
  operators
    functions vs, 398–399
    insertion and extraction, 399
    move constructor and operator, 407–410
    overloading assignment operator, 401–405
    overloading subscript operator, 406–407
    shallow copies, 399–400
semesterHours, 221, 224, 278, 288
set, 280
setter functions, 280–282
ShallowCopy sample program, 333–334
shared pointer, 270–273
shared_ptr, 463
short-circuit evaluation, 77–78
showTheDetails() function, 110
signature, 112
signed version of integers, 40
simpleDivision() function, 447–448
single-bit operators, 80–81
sizeof(char), 355
sizeof(name), 355
sizeText string, 108
size_type, 241

smart pointers, 270–273, 298–299
  benefits of, 161
  definition of, 161
  example of, 162
  make_shared() function, 162
  pMyString pointer, 161, 162
  pointing to array, 163
  shared pointer, 162, 163
  unique pointer, 164
source file, definition of, 10
spaceship operator, 78–79
span, 132–134
special characters, 44–45
  printing, 63, 64
spendMoney(), 277
square() functions, 203
s.semesterHours, 283
stack unwinding, 439
Standard Template Library (STL), 392
statements
  branch statements, 84–86
  declaration, 29
  I/O statements, 29, 30
  println statements, 29–30
  print statements, 29–30
  whitespace, 28
static_assert() function, 195, 196
static int, 346
static keyword, 345, 347, 355
static member functions, 378
static members
  defined, 347
  static data members, 347
  counting objects, 351
  flagging actions, 351
  need for, 347–348
  providing space for head pointer, 351
  referencing, 349–351
  using, 348–349
  static member functions, 352–355
  this keyword, 356
static objects
  defined, 317
  rules for order of construction, 317–319
static variables, 345, 352
staticVariable variable, 120, 121
std::filesystem namespace, 415
stem(), 416
STLString program, 241–243
storing logical values, 73–74
strcat() function, 142
strcmp() function, 142
strcpy() function, 142, 176
streaming, 62, 69
str ...() functions, 142
string class, 235, 237
  methods of, 239–240
  overloaded operators in, 241
string-handling functions, 146–147
string.insert(), 243
string manipulation, pointer variables, 172–173
string() method, 416
strings
  array of characters and, 236
  changing numbers to, 249–250

container, 236–238
functions, 238–245
gaining new view of, 245–246
manipulation, 137–139, 142
string_view, 245–246
strncat() functions, 142
strncpy() functions, 142
struct keyword, 231–234
structure binding, 393
Student, 277–280, 282–285, 288–290, 293, 299
subclasses, 366
of abstract classes, 391
constructing, 367–368
destructing, 368–369
inheriting constructors, 369
subscript operator ([ ]), 406
subscript operator, overloading, 406–407
sumArray() function, 127, 128
sum variable, 51
switch statement, 98–100
sz prefix, 137

T

\t tab character, 45
Teacher, 284–286, 293
ternary (?:) operator, 398
testFunction() function, 193
testing, code, 470
third-party code, 469
third-party libraries, 469
this keyword, 356
three-way comparison operator, 78–79
throw keyword, 436–437, 442–446
to_string() function, 249–250
to_underlying() function, 192
trivial relocatability, 344
truncation, 35–36
floating-point numbers, 37
truth table
for AND operator, 80
for OR operator, 81
for XOR operator, 81
try keyword, 436–437, 439, 441
tuple, 392–393
TutorPair constructor, 293–294, 297
typedef keyword, 196–197

U

Ubuntu Linux, 11
Code::Blocks installation, 16
gcc compiler installation, 15
unary operators, 52, 55–57
decrement (––) operators, 56
increment (++) operators, 56
minus-minus operator (––), 56
minus unary operator (–), 55
plus operator, 55
plus-plus operator (++), 56
Unicode Transformation Format (UTF)
UTF-8, 46, 47
UTF-16, 46
UTF-32, 46
unique_ptr function, 462, 463
unpexected type, 450–451
unsigned version of integers, 40

unwinding the stack, 439
user-created assignment operator, 405
`using` keyword, 196–197, 200
`using` statement, 415

V

`value()`, 448–449, 451
variables
  advantages and limitations, 39–40
  automatic, 129
  character, 41
  character string, 41
  constants, 42–43
  declaring, 160
    overview, 34
    types, 39–45
  definition of, 29, 33
  double precision
    accuracy, 38
    range, 39
  floating-point
    accuracy, 38–39
    calculation speed, 38
    counting, 38
    definition of, 37
    range, 39
  integer
    accuracy, 39
    definition of, 35
    range, 39
    truncation, 35, 37
  memory allocation, 160–161
  mixed-mode expressions, 47, 48
  naming, 48
  nonprintable characters, 44, 45
  printing variables, 65–66
  range of numeric types, 43–44
  scope, 121
  signed and unsigned versions, 40
  special characters, 44–45
  static, 345
  visible variables, 120–121
vectors, 461
version control, 472
`virtual`, 463–464
virtual base classes, 430
virtual destructors, 463–464
virtual inheritance, 424–430
`VirtualInheritance` sample program, 428–430, 433
`virtual` keyword, 375–376, 428–429
virtual member functions
  constructors, 378
  declaring, 375–376
  declaring functions as not overrideable, 377
  destructors, 379–380
  overriding base class functions, 376–378
  pure, 389–390
  `static`, 378
visible variables, 120–121
Visual Studio, 43–44, 184
`void` function, 106, 107

W

warning messages, enabling all, 458
`wchar_t` (wide character) variables, 46
weakly typed languages, 35
`WhileDemo` sample program, 87–89
`while` loops, 86–88
  autodecrement feature, 88–89
  `do...while` loop, 88
whitespace, 28, 85
wide character (`wchar_t`) variables, 46
wide string-handling functions, 146–147
windowed programming, 11
workspace file, 22
`wstring` class, 147

X

Xcode development package, installing, 17
Xcode installation, 17
XOR operators, 81

Z

zero-terminated string, 137