# INDEX

## A

abort function, 162, 177, 232
abort_handler_s function, 164–165
ABS macro, 54–56
addition (+) operator, 25
additive operators, 83–84
address, 14
AddressSanitizer, 132, 253–257
    compiler flags used with, 255
address space layout randomization (ASLR), 239
Advanced Encryption Standard (AES), 56
aligned_alloc function, 120
alignment requirements, 41–42
alignof operator, 92
alloca function, 128–129
allocated storage duration, 35, 116
/analyze flag, 241
Annex K bounds-checking interfaces
    gets_s function, 161–162
    runtime constraints, 163–164
    strcpy_s function, 162–163
application programming interface (API), 168, 216
argument checking, 156–157
arithmetic types, 19–22, 47
    conversion, 64–72
    enumeration, 21
    floating-point, 21–22, 59–64
    integer, 48–59
    operators, 83–85
    pointer, 94–96
arm_missile function, 19
ARRAY_SIZE identifier, 203
array types, 25–27
ASCII, 138
assertions
    runtime, 232–234
    static, 230–232
assert macro, 232
assignInterestRate function, 103–104
assignment (=) operator, 74
associativity, 78–80
ATOMIC_VAR_INIT macro, 207
attributes, 44–45, 259–260
automatic storage duration, 15
auto storage-class specifier, 38–39

## B

backslash (\), 143
basic execution character set, 19
basic multilingual plane (BMP), 139, 146
big-endian ordering, 191
binary constants, 58
binary resources, embedding, 209–210
binary streams, 172
    reading from and writing to, 188–191
*bin* directory, 225–227
bit and byte utilities, 264–265
_BitInt type, 56–57
bit-precise constants, 58
bit-precise integers, 20, 56–59
bitwise operators
    bitwise AND (&), 87
    bitwise exclusive OR (^), 88
    bitwise inclusive OR (|), 88–89
    complement, 85–86
    shift, 86–87
block, 14. *See also* compound statements
block scope, 34
Boeing 787, 51
Boolean type, 18–19
break statement, 111–112
buffering, 169–170

buffer overflows, 106, 123, 152, 159, 160, 164–165, 188, 239, 240
byte-oriented stream, 171
bytes, 154

## C

C, xxv–xxvi
  history and development of, xxiii–xxv
  getting started with, 1–11
call-by-value language, 16
`calloc` function, 120–121
`cardinal_points` enumeration, 21
cast operators, 90–91
`cc` command, 4
central processing units (CPUs), 41, 85
`char16_t` type, 142
`char32_t` type, 142
characters
  ASCII, 138
  code points, 138
  constants, 142–143
  conversion of, 146–149
  data types, 140–142
  escape sequences, 143–144
  execution character set, 140
  Linux, 144–145
  literals, 142–143
  narrow, 140–142, 146
  reading and writing, 177–180
  source character set, 140
  Unicode, 138–140
  wide, 19, 140–142, 146
character string literal, 150–152
`char` type, 19, 140–141
Clang, 44–45, 93, 102, 104, 129, 131, 134, 145, 196, 198, 203, 206, 226, 252, 255
  flags, 235–240
  installation, 8
  predefined macros list, 210
`clearerr` function, 169
`clear_stdin` function, 231
`close` function, 177
code
  building, 225–227
  page, 138
  points, 138, 154
  reuse, 215
  units, 139, 154
coercion, 65
cohesion, 214–215
collection, 216
`collection_type` identifier, 217
comma (`,`) operator, 94
`CommandLineToArgvW` function, 145
common extensions, 11
compilation process, preprocessor, 196
compilers, 7–8
compiler settings and flags, 234–235
  GCC and Clang, 235–240
  Visual C++, 240–241
compiling and running a program, 3–5
complement operators, 85–86
complex types, 22
componentization, 213–218
  code reuse, 215
  cohesion, 214–215
  coupling, 214–215
  data abstraction, 215–216
  opaque types, 217–218
compound assignment operators, 93–94
compound statements, 14, 98–99
conditional inclusion, 198–202
  generating diagnostics, 200–201
  header guards, 201–202
conditional (`? :`) operator, 91–92
considered behavior, 10
constants
  character, 142–143
  floating, 64
  integer, 57–59
    binary, 58
    decimal, 57
    hexadecimal, 57–58
    octal, 57
`constexpr` storage-class specifier, 37–38
`const` qualifier, 32
`continue` statement, 111
control flow, 97–113. *See also* jump statements; selection statements
  compound statement, 14, 98–99
  expression statements, 97–98
  iteration statements, 105–108

control flow guard (CFG), 240–241
controlling expression, 99–105
conversion
  arithmetic, 64–72
  characters, 146–149
  floating-type demotions, 71
  floating-type to integer-type and vice versa, 71
  implicit, 69–70
  integer, 65–67, 70–71
  safe conversion, 70–71
  specifiers, 187
  usual arithmetic conversions, 67–69
`convert_arg` function, 222
`convert_cmd_line_args` function, 222
`copy_process` function, 110
coupling, 214–215
C standard library, 147–149

## D

dangling pointers, dealing with, 125–126
data abstraction, 215–216
debugging, 241–245
decimal constants, 57
decimal floating types, 22
declarations, 74
declarations, integer, 49
decrement (`--`) operator, 77
`defined` operator, 199
`#define` preprocessing directive, 202–203
defragmentation, 117
derived types
  array, 25–27
  function, 22–23
  pointer, 23–25
  structure, 27–28
  union, 28–29
diagnostics, generating, 200–201
`dmalloc` (debug memory allocation), 132–134
  library, 252
`double` type, 59–62
`do...while` loop, 231
`do...while` statement, 106–107
`dup_string` function, 233
dynamically allocated memory, 115
dynamic analysis, 252–253
dynamic library, 218–219
  debugging, 132–135
  flexible array members, 127–128
  memory management, 117–126
  safety-critical systems, 134–135
  storage duration, 116–117
  using, 117

## E

editors, 6–7
8-bit representation, 51
`else` statement, 199
`#embed` preprocessor directive, 209
endianness, 28, 191–193
`#endif` preprocessing directive, 199
enumerations, 21, 261
environment variables, 164
equal (`==`) operator, 70
`#error` preprocessing directive, 201
escape sequences, 143–144
Euclidean division, 84
evaluation, 75–76
  indeterminately sequenced, 81
  order of, 80–82
  unsequenced, 81
executables, 218–219
execution character set, 140
`EXIT_SUCCESS` macro, 2–3
`EXPECT_STREQ` assertion, 247
explicit undefined behaviors, 10
expressions, 73
  evaluations, 75–76
  function invocation, 76–77
  simple assignment, 74–75
  statement, 97–98
Extended ASCII, 138
extended grapheme cluster, 154–155
extended integer types, 20
`extern "C"` declaration, 247
`extern` specifier, 37
`extern` storage class specifier, 220

## F

`fclose` function, 176–177, 181, 188
`feof` function, 169
`ferror` function, 169
`fgetc` function, 178

fgetpos function, 181
field-programmable gate arrays (FPGAs), 56
file access modes, 174
file descriptor, 174
file inclusion, preprocessor, 197–198
FILE object, 168, 174, 177
file pointers, 168
file scope, 34
files, I/O
    closing, 176–177
    creating, 172–176
    opening, 172–176
file status flags, 175
flag argument, 55
flags, 235–240
flexible array members, 127–128
floating-point
    arithmetic, 21–22, 62
    C model, 60–62
    constants, 64
    encoding, 59–60
    types, 59–60
    values, 62–64
float type, 59–60
flooring division, 84
flushing, 170, 180
fopen function, 172–174, 181
format string, 185
formatted output, 5
formatted text streams, reading, 184–188
for statement, 107–108
_FORTIFY_SOURCE macro, 239
fpclassify macro, 63–64
FPGAs, 56
-fpic flag, 239
-fpie flag, 239
fread function, 188–191
free_aligned_sized function, 124–126
free function, 123–224
free_sized function, 124
freestanding environment, 2
fseek function, 173, 180–181
fsetpos function, 173, 181
-fstack-protector-strong option, 240
functions, 2, 14
    code reuse, 215
    declarator, 22
    definition, 23
    designator, 76–77
    invocation, 76–77
    objects and, 14
    prototype, 23
    return values, 4–5
    scope, 34
    type, 22–23
    variadic, 76
fwrite function, 188–191

## G

GCC. *See* GNU Compiler Collection
generic selection expression, 207
getc function, 178
getchar function, 178
get_error function, 246
get_file_size function, 181
get_password function, 159
gets function, 160–161
gets_s function, 161–162
-glevel flag, 236–237
GNU Compiler Collection (GCC), xxv–xxvi, 8, 21, 93, 102–104, 129, 131, 133, 134, 144, 196, 198, 203, 206, 210, 226, 252, 253
    compiler and linker flags, 235–240
Google Test, 245–246
goto statement, 109–110
/guard:cf flag, 240–241

## H

__has_c_attribute operator, 262
__has_include operator, 262
__has_include preprocessor operator, 198
header files, 2, 216
header guards, 201–202
heap manager, 116–117
hexadecimal constants, 57–58
hidden scope, 34

## I

identifier, 14
IDEs, 6–7
IEEE floating-point support, 265
if...else ladder, 102

if statement, 99–102, 199
ignore_handler_s function, 164–165
implementation-defined behaviors, 9
implicit conversion, 65, 69–70
implicit undefined behavior, 10
#include preprocessor directive, 2
incomplete array type, 127
increment (++) operator, 77
indeterminately sequenced evaluation, 81
indirection (*) operator, 16–17, 24–25
infinite loop, 106
initializer, 74
inner scope, 34
input/output (I/O), 2, 33, 146, 167–168
integers
  bit-precise, 20, 56–57
  constant expressions, 260–261
  constants, 57–59
  conversion rank, 65–66
  conversions, 70–71
  declarations, 49
  overflow, 54–56
  padding, 48
  precision, 48
  promotions, 66–67
  ranges, 48
  signed, 52–56
  unsigned, 49–52
  width, 48
integrated development environments (IDEs), 6–7
intentional behavior, 10
Internet Protocol (IP), 191
int type, 141
I/O. *See* input/output
is_prime function, 224

## J

jump statements
  break, 111–112
  continue, 111
  goto, 109–110
  return, 112–113

## K

K&R C functions, 262
keywords, 260
KnownError test case, 249
Knuth, Donald, 116, 214

## L

labels, 34
libiconv function, 149
libraries, 218
lifetime, determining, 15
lines, reading and writing, 177–180
linkage, 219–221
link phase, 218
Linux, 144–145
literals
  character, 142–143
  string, 150–152
little-endian ordering, 191
locale, 140
locale-specific behavior, 11
locator value, 74
logical AND (&&) operator, 89–90
logical negation (!) operator, 83
logical OR (||) operator, 89
long double type, 59–62

## M

macro
  definitions, 202–211
  embedded binary resources, 209–210
  generic selection expression as, 208
  predefined, 210–211
  replacement, 205–207
  type-generic, 207–209
    with automatic type inference, 209
  undefining, 204–205
  unsafe expansion, 206
main entry point, 145–146
main function, 2, 17, 40, 225
malloc function, 118–120
matrix, 26
max function, 76
mbrtoc16 function, 148
mbsrtowcs function, 147
mbstowcs function, 147
mbtowc function, 147
memccpy function, 157–159, 243–244

memcpy function, 157
__memmove_avx_unaligned_erms function, 243
memory
  allocating without declaring type, 118–119
  leaks, 117
    avoiding, 121–122
  management of, 117–126
  manager, 116–117
  reading uninitialized memory, 119–120
  states of, 126–127
memset_explicit function, 159–160
memset function, 159–160
memset_s function, 159–160
Miller-Rabin primality test, 224
MultiByteToWideChar function, 149
multiplicative operators, 84

## N

NAME macro, 205
namespace, 30
narrow characters, 140–142, 146
narrow string, 149
NDEBUG macro, 233
nesting, 34
not-a-number (NaN), 63
NULL macro, 24
null pointer, calling realloc with, 122
nullptr pointer, 126
num_args parameter, 223

## O

objects, 13–14. *See also* types
  storage
    class, 36–39
    duration, 35–36
octal constants, 57
-O flag, 235–236
opaque types, 217–218
open file description, 174
open function, 174–176
operating system (OS), 116
operators, 73
  alignof, 92
  arithmetic, 83–85
  associativity, 78–80
  bitwise, 85–89
  cast, 90–91
  comma (,), 94
  compound assignment, 93–94
  conditional (? :), 91–92
  decrement (--), 77
  increment (++), 77
  logical, 89–90
  order of evaluation, 80–82
  postfix, 77
  precedence, 78–80
  prefix, 77
  relational, 93
  sizeof, 82–83
order of operations, 78
original equipment manufacturer (OEM), 145–146
outer scope, 34
overflow, integer, 54–56

## P

padding, 48
parameters, 16
pass-by-value language, 16
-pedantic flag, 238
pedantic mode, 11
/permissive- flag, 241
-pie flag, 239
planes, 138–139
++1 operation, 77
pointer arithmetic, 94–96
pointers, 14, 23–25
portability, 9–11
  common extensions, 11
  implementation-defined behaviors, 9
  locale-specific behavior, 11
  undefined behavior, 10–11
  unspecified behaviors, 10
Portable Operating System Interface (POSIX), 24, 164–165, 183–184, 189
postfix operators, 77
precedence, operator, 78–80
precision, 48
predefined macros, 210–211
predefined streams, 170–171
predicate. *See* assertions

prefix operators, 77
preprocessing directives, 196
preprocessor, 195
  compilation process, 196
  conditional inclusion, 198–202
  file inclusion, 197–198
  macro definitions, 202–211
  translation phases, 196
prime number, 221
printerr function, 129, 130
print_error function, 130, 253–254
printf function, 5
print_help function, 221
program structure
  building code, 225–227
  componentization, 213–218
  executables, 218–219
  linkage, 219–221
  simple program, 221–225
promotions, integer, 66–67
pthread.h header, 200
public interface, 215–216
putc function, 178
puts function, 2, 5, 178

**Q**

qualified types, 31–34

**R**

random-access memory (RAM), 184
real floating types, 22
reallocarray function, 117–118, 123
realloc function, 121–122
rec.signame, 186, 188, 190
referenced type, 23
register storage-class specifier, 38
relational operators, 93
representable value, 48
restrict-qualified pointer, 33–34
return statement, 112–113
return values, function, 4–5
rewind function, 173, 182–183
rmdir function, 184
RUN_ALL_TESTS macro, 247
runtime analysis, 252–253
runtime assertions, 232–234
runtime constraints, 163–164
rvalue (right operand), 74

**S**

safe conversion, 70–71
safety-critical systems, 134–135
scope, 34–35
  block, 34
  file, 34
  function, 34
  function prototype, 34
/sdl flag, 241
Secure Hash Algorithm (SHA), 56
selection statements
  if, 99–102
  switch, 102–104
sequence points, 81–82
set_constraint_handler_s function, 163
setlocale function, 148
7-bit ASCII, 138
shadowed scope, 34
-shared flag, 239
shift operations, 86–87
show_classification function, 63
side effects, 55, 76
signed char type, 19
signed integers, 20
sign extension, 70
  integer overflow, 54–56
  representation, 52–54
simple assignment, 74–75
simple program, structuring, 221–225
sin function, 207
single quote ('), 143
sizeof operator, 82–83, 131, 154
sizeof(size++) operand, 131
small types, 66–67
software development kit (SDK), 146
source character set, 140
source files, 216
spaghetti code, 109
standard error stream (stderr), 171
standard input stream (stdin), 170
standard output stream (stdout), 170
states, memory, 126–127

static
  analysis, 251–252
  assertion, 230–232
  keyword, 220
  library, 218
static storage-class specifier, 36–37
__STDC_ENDIAN_BIG__ macro, 192
__STDC_ENDIAN_LITTLE__ macro, 191
__STDC_ENDIAN_NATIVE__ macro, 192
/std:clatest flag, 241
-std= flag, 238
storage
  class, 36–39
  duration, 35–36
    heap manager, 116–117
    memory manager, 116–117
    using dynamically allocated memory, 117
  other forms of, 128–132
strcpy function, 155–156
strcpy_s function, 162–163
streams, I/O
  binary, 172
  buffering, 169–170
  error and end-of-file indicators, 168–169
  orientation, 171
  predefined, 170–171
  text, 172
strerrorlen_s function, 246
strerror_s function, 246, 249
strictly conforming programs, 9
string-handling functions, 152–165
stringizing, 205
strings, 137–138, 149–152
strlen function, 154–155
strndup function, 164
structure member (.) operator, 27
structure pointer (->) operator, 27
structure type (struct), 27–28
subnormal numbers, 62
subscript ([]) operator, 25
substatement, 99–102
supplementary characters, 139
surrogates, 139
swap function, 16–17
switch statement, 102–104

**T**

tags, 29–31
tempfile, 170–171
temporary files, 184
test suite, 246
text stream, 172
thread_local storage-class specifier, 37
threads.h header, 200
thread storage duration, 35, 37
time-of-check to time-of-use (ToCToU), 33
token pasting, 206
translation phases, 196
translation unit, 196
Transmission Control Protocol (TCP), 191
truncating division, 84
type, 14
typedef storage-class specifier, 38
type-generic macros, 207–209
type inference, 261
typeof operators, 39–40, 262–264
  integer types and representation, 263–264
  K&R C functions, 262
  preprocessor, 262
typeof_unqual operator, 39–40
types
  object
    arithmetic, 19–22
    Boolean, 18–19
    character, 19
    void, 22
  definitions, 26–27
  derived, 22–29
    array, 25–27
    function, 22–23
    pointer, 23–25
    structure, 27–28
    union, 28–29
  qualifiers, 31–34
    const, 32
    restrict, 33–34
    volatile, 32–33
  variably modified, 42–44

## U

Ubuntu Linux, 246
UINT_MAX expression, 50–52
unary & (address-of) operator, 17
unary + and - operators, 83
undefined behavior, portability and, 10–11
Unicode scalar value, 139
Unicode Standard (Unicode), 138–140
Unicode transformation formats (UTFs), 139
uninitialized memory, 119–120
union types, 28–29
unit testing, 245–251
Universal Serial Bus (USB) ports, 168
unreachable function-like macro, 264
unsequenced evaluations, 81
unsigned char type, 19
unsigned integers, 20
  representation, 49–50
  wraparound, 50–52
unsigned-preserving approach, 67
unspecified behaviors, 10
Urban, Reini, 246
User Datagram Protocol (UDP), 191
usual arithmetic conversions, 67–69

## V

value computation, 75–76
value-preserving approach, 67
valueReturnedIfTrue operand, 91–92
values, swapping, 15–18
variable-length arrays (VLAs), 129–132, 260
variables, 14
  declaring, 14–18
variably modified types (VMTs), 42–44
Visual C++, 21, 91, 139, 142, 145, 165, 196, 203, 240–241, 252, 255
Visual Studio Code (VS Code), 6–7, 242
Visual Studio IDE, Microsoft, 6, 8
void type, 22
volatile-qualified type, 32–33
vstrcat function, 241

## W

-Wall flag, 237–238
wchar_t type, 141–142
-Wconversion flag, 238
wcslen function, 154
wcsrtombs function, 147
wcstombs function, 147
-Werror flag, 238
-Wextra flag, 237–238
while statement, 105–106
wide characters, 19, 140–142, 146
wide-oriented stream, 171
wide string, 150
width, integer, 48
-WI flag, 239
Win32 conversion APIs, 149
Windows, 145–146
-Wl,-z,noexecstack linker option, 239–240
wmain entry point, 145–146
word, 41
wraparound, 50–52