

# Index

## Special characters and numbers

- ( ) (parentheses)  
function syntax, 13  
mismatched, debugging, 513
- , (comma), number punctuation, 56
- . (dot), current directory shortcut, 602
- .. (dot dot), parent directory  
shortcut, 602
- ?: (question mark colon), ternary  
operator, 212–213
- & (ampersand)  
binary AND, 391–394  
unary address operator  
initializing pointers, 355, 358, 365  
in `scanf()` function, 358, 365
- \* (asterisk)  
multiplication operator  
definition, 110  
floating-point multiplication, 50  
unary pointer operator  
initializing pointers, 364  
modifying variables, 376–380  
previewing RAM contents, 374–376  
requirements for, 361  
syntax, 360
- \*= (asterisk equal), shortcut operator, 121
- \*\* (asterisks), pointer-pointer prefix, 443
- ^ (caret)  
binary XOR (exclusive OR), 399–400  
exponent symbol, 167–169
- \$ (dollar sign), representing hexadecimal  
numbers, 154
- \$? (dollar sign question mark), Unix exit  
status, 545–546
- “ (double quotes)  
in C statement syntax, 16  
escape sequence for, 41  
as literals in C statements, 41
- = (equal sign), assignment operator,  
23, 54
- == (equal signs), comparison operator  
definition, 130  
string comparisons, 247
- < (left angle bracket), input  
redirection, 527
- << (left angle brackets), shift bits left,  
388, 390–391
- = (minus equal), shortcut operator, 121
- (minus sign)  
subtraction, 110  
unary negative operator, 110
- (minus signs), decrement operator,  
115, 121
- % (percent sign), modulo operator,  
185–186
- %% (percent signs), conversion  
character, 771
- += (plus equal), shortcut operator, 121
- + (plus sign)  
addition operator, 110  
unary positive operator, 110
- ++ (plus signs), increment operator,  
115, 121
- # (pound sign), precompiler directives, 25
- > (right angle bracket), output  
redirection, 528
- >> (right angle brackets), shift bits right,  
388–390
- ‘ (single quote)  
in character variables, 62  
escape sequences, 63  
as literal character, 63

## 774 C All-in-One Desk Reference For Dummies

/ (slash), division operator  
 definition, 110  
 floating-point division, 51  
 /\* ... \*/ (slash asterisk...), comment  
 delimiters, 36–37  
 /= (slash equal), shortcut operator, 121  
 // (slashes), comment delimiter, 40  
 \_ (underscore), in functions, 295  
 | (vertical bar)  
 binary OR, 394–395, 397–398  
 piping output, 528–530  
 [ ] (square brackets), array brackets,  
 411–413  
 ; (semicolon)  
 C statements, 15–16, 24  
 constants, 104  
 function input, 304  
 precompiler directives, 25  
 { } (braces)  
 function syntax, 13  
 if statement syntax, 82  
 ~ (tilde), one's complement, 400–402  
 \0, NULL character, 248–250  
 2-dimensional arrays, 236–239  
 3-dimensional arrays, 239–240, 259–261

### A

Abigor, 473  
 abnormal ending, 322–323  
 abort() (abort program) function,  
 322–323  
 abs() (absolute values) function,  
 177–180  
 addition  
 + (plus sign), addition operator, 110  
 ++ (plus signs), increment operator,  
 115, 121  
 immediate, 49  
 incrementing by more than one,  
 119–120  
 incrementing by one, 115–119  
 incrementing values and addresses,  
 408–410  
 for loops, 91  
 pre/post-incrementing, 117–119  
 addresses (storage location). *See also*  
 pointer arrays; pointers; RAM  
 & (ampersand), unary address  
 operator  
 initializing pointers, 355, 358, 365  
 in scanf() function, 358, 365  
 arrays, getting, 365–366  
 arrays, incrementing, 408–410  
 arrays, pointers to, 364–366  
 definition, 357  
 pointers, incrementing, 408–410  
 RAM (random access memory), 357  
 variables, description, 355–358  
 variables, getting, 355–358  
 variables, pointers to, 360–364  
 Algol, 10  
 Algorithms in C, 234  
 allocating RAM. *See also* malloc()  
 function; realloc() function  
 arrays, 480–484  
 structures, 480–484  
 variables, 351–352  
 alphabetical data. *See* text  
 ampersand (&)  
 binary AND, 391–394  
 unary address operator  
 initializing pointers, 355, 358, 365  
 in scanf() function, 358, 365  
 Analytical Engine, 10  
 AND operator, 127, 129–130  
 Apple, online resources, 670  
 ar tool, 515  
 argc argument, 538–539

- arguments
    - command line
      - parsing, 538–541
      - returning, 542–546
      - testing for, 540–541
      - viewing, 539–540
    - in functions, 309
    - main() function, 13
  - argv argument, 538
  - arithmetic. *See* mathematical operations; numbers
  - array notation
    - pointer arrays, 437–439, 441–443
    - pointers, and functions, 461–462
    - replacing with pointers, 405, 406–408, 411–415
  - arrays. *See also* pointer arrays
    - address pointers, 364–366
    - definition, 223
    - element numbering, 225
    - and functions
      - modifying, 463–464
      - passing to functions, 460–464
      - returning from functions, 464–466
      - size, determining, 462
    - input to functions, 310
    - multidimensional
      - 2 dimensions, 236–239, 236–239
      - 3 dimensions, 239–240, 259–261
      - structure of, 236
    - one dimensional
      - creating, 224–225
      - declaring empty, 226–228
      - refilling, 228–229
      - using, 225–226
    - reading from disk, 564–566
    - replacing with pointers
      - [ ] (square brackets), array brackets, 411–413
      - array notation, eliminating, 405, 406–408, 411–415
      - filling arrays with pointers, 405–406
      - incrementing values and addresses, 408–410
      - pointer variables, table of, 410
      - representing strings, 247
      - size, changing, 227, 480–484
      - size, getting, 353–355
      - sorting, 229–235
      - string representation, 247
      - of strings, 257–263
      - of structures, 283–289
      - writing to disk, 564
  - ASCII
    - code table, 681–684
    - converting to numeric values, 74–75
    - definition, 65
  - assignment operators, 764–765
  - asterisk (\*)
    - multiplication operator
      - definition, 110
      - floating-point multiplication, 50
    - unary pointer operator
      - initializing pointers, 364
      - modifying variables, 376–380
      - previewing RAM contents, 374–376
      - requirements for, 361
      - syntax, 360
  - asterisk equal (\*=), shortcut operator, 121
  - asterisks (\*\*), pointer-pointer prefix, 443
  - atexit() (at exit) function, 325–327
  - atof() (ASCII-to-float conversion)
    - function, 74–75, 78
  - atoi() (ASCII-to-integer conversion)
    - function, 74–75, 78
  - AT&T Bell Labs, 10
  - auto keyword, 335
- 
- B**
- 
- B language, 10
  - Baal, 473
  - Babbage, Charles, 10
  - \0, NULL character, 248–250

## 776 C All-in-One Desk Reference For Dummies

- banking, sample programs, 630–641, 648–654
  - base 2 (binary) numbering system. *See also* binary operations
    - counting by twos, 382–384
    - definition, 151
    - displaying binary values
      - in binary, 395–398
      - in hexadecimal, 385–387
  - base 8 (octal) numbering system, 156–157
  - base 10 (decimal) numbering system, 150
  - base 16 (hexadecimal) numbering system
    - case sensitivity, 154
    - displaying, 151, 163
    - escape sequences, 155–156
    - identifying, 151
    - overview, 151–153
    - representing, 151, 154, 163
  - BCPL language, 10
  - Beelzebub, 473
  - Belial, 473
  - binary files *versus* text, 554–559
  - binary (base 2) numbering system
    - counting by twos, 382–384
    - definition, 151
    - displaying binary values
      - in binary, 395–398
      - in hexadecimal, 385–387
  - binary operations. *See also* base 2 (binary) numbering system
    - bits
      - definition, 382
      - shifting, 388–391
    - bitwise operators
      - << (left angle brackets), shift left, 388, 390–391
      - >> (right angle brackets), shift right, 388–390
      - list of, 387
    - bytes, definition, 385
    - logical operations, 391–395
    - logical operators
      - & (ampersand), AND, 391–394
      - ^ (caret), XOR (exclusive OR), 399–400
      - | (vertical bar), OR, 394–395, 397–398
    - long words, definition, 385
    - words, definition, 385
  - BinString() (binary string) function, 395–397
  - bits
    - definition, 382
    - shifting, 388–391
  - bitwise operators
    - << (left angle brackets), shift left, 388, 390–391
    - >> (right angle brackets), shift right, 388–390
    - list of, 387
  - blanks, in functions, 295
  - books and publications
    - Algorithms in C*, 234
    - C For Dummies*, 612
    - The New Hacker's Dictionary*, 511
  - bounds checking, 73
  - braces ({ })
    - function syntax, 13
    - if statement syntax, 82
  - break keyword, 96
  - breaking out of loops, 96
  - buffers, 556
  - bugs, 42, 44–45. *See also* debugging
  - bytes, definition, 385
- 
- ### C
- 
- C++
    - history of, 10
    - name origin, 117
    - reserved keywords, 763
  - %c, conversion character, 771

- .C file extension, 25
- C For Dummies*, 612
- C language. *See also* program skeleton
  - derivative languages, 1–2
  - history of, 9–10
  - syntax, 24
- c option, 500
- calling
  - functions, 295–296
  - main() function, 13
- capitalization. *See* case
- caret (^)
  - binary XOR (exclusive OR), 399–400
  - exponent symbol, 167–169
- case, converting to
  - lower, 266, 269–270
  - upper, 266, 268–269
- case sensitivity
  - editors, 25
  - filenames, 549
  - hexadecimal numbers, 154
  - strings, 251, 254
- cbrt() (cube root) function, 169–170
- .CC file extension, 25
- char variable
  - declaring, 64
  - definition, 14, 137
  - as integer value, 65–67
  - single character, 62–63
  - size limitation, 67
- character data. *See* strings; text
- character variables
  - declaring, 64
  - definition, 14, 137
  - as integer value, 65–67
  - single character, 62–63
  - size limitation, 67
- chdir() (change directory) function, 595–597
- Clarke, Arthur C., 98–99
- closedir() (close directory) function, 589–590
- closing
  - directories, 588–590
  - files, 550–551
- COBOL, 10
- Code Warrior compiler, 670
- color coding, vim editor, 63, 674
- comma (,), number punctuation, 56
- command line
  - counting arguments, 539
  - exit status, returning, 542–546
  - parsing arguments from, 538–541
  - returning arguments, 542–546
  - simulating in programs, 541–542
  - testing for arguments, 540–541
  - viewing arguments, 539–540
- commenting out code, 38
- comparing code. *See* decision making
- comparing strings, 247, 253–254
- compiler errors, 42
- compilers. *See also* GCC compiler
  - Code Warrior, 670
  - error checking, 27
  - file management, 670–672
  - history of, 10
  - Macintosh, 670
  - precompiler directives, 25
  - purpose of, 20
  - setting up, 670–672
  - Windows, 670–671
- compiling. *See also* linking
  - with linking, 675–680
  - without linking, 500
  - multiple modules, 500
  - object code, 25–29
  - programs with multiple modules, 498–501
- complement, 402
- computer languages, 20. *See also* *specific languages*

## 778 C All-in-One Desk Reference For Dummies

- computer memory. *See also* addresses (storage location)
- allocating *See also* `malloc()` function; `realloc()` function
    - arrays, 480–484
    - structures, 480–484
    - variables, 351–352
  - contents of, 350–351
  - definition, 349–350
  - freeing, 478–480
  - previewing contents, 374–376
  - saving space, 431–432
  - size, arrays, 353–355
  - size, variables, 352–353
  - `sizeof` operator, 353
  - variable addresses, 355–358
- concatenating strings, 254–256
- conditional operator, 212–213
- conditional statements (decision making). *See also* `goto` statement
- `?:` (question mark colon), ternary operator, 212–213
  - conditional operator, 212–213
  - `else if` statement, 84–87, 205–206
  - `else` statement, 83–84
  - falling through, 209–210
  - `if` statement
    - character comparisons, 80–82
    - equality comparisons, 80–81
    - greater-than comparisons, 81–82
    - less-than comparisons, 81–82
    - overview, 79–80
  - logical comparisons
    - `==` (equal signs), comparison operator, 130
    - AND operator, 127, 129–130
    - OR operator, 127–128
    - order of precedence, 132
    - Yes-or-No puzzle, 123–127, 131
  - multiple decisions, 84–87, 205–209
  - order of operations, 85–87
  - switch-case statement, 205–209
    - switch-case `while(!done)` technique, 211–212
  - `const` keyword, 106, 335
  - constants. *See also* values; variables
    - declaring, 104–105, 335
    - overview, 103–104
    - sample program, 101–103, 105
    - variables instead of, 103
  - `continue` keyword, 202–203
  - conversion characters (placeholders), 771. *See also* specific characters
    - integers, 48
    - and keyboard input, 77
    - list of, 771
  - Coordinated Universal Time (UTC), 487
  - copying
    - files, 611
    - strings, 251–252
    - structure elements, 287–289
  - `cos()` (cosine of an angle) function, 173–177
  - counting bases, 150. *See also* numbering systems
  - counting by letters, 98
  - .CPP file extension, 25
  - craps game, 213–216, 312–315. *See also* rolling dice
  - `ctime()` (current time) function, 490–492
  - CTYPE.H character change functions, 266
  - CTYPE.H test functions, 266
  - current directory, shortcut, 602
- 
- ### D
- 
- `%d`, conversion character
    - definition, 771
    - example, 48
    - output value width, setting, 162–163

- data types
  - list of, 14, 136–137, 767
  - mixing, 342–346
  - redeclaring, 146–148
  - treating one as another, 146–148
- data types, variable data
  - changing temporarily
    - creating structure variables, 332–333
    - linked lists, 333–334
    - overview, 146–148
    - renaming variables, 331–332
    - structures, 279
- databases. *See also* linked lists; structures
  - converting from arrays to structures, 284–285
  - on disk, 582–584
  - records, 273–274
  - sample program, 630–641, 648–658
  - Wizard of Oz* example, 272–277, 284–285
  - writing to disk, 582–584
- date and time
  - element structure, 492–493
  - the epoch, 487–488
  - GMT (Greenwich Mean Time), 487
  - Gregorian calendar, 486–487
  - Julian dates, 486
  - Modified Julian Day, 486
  - time, calculating a pause, 494–496
  - time, getting
    - current day, 491–492
    - as epoch value, 488–490
    - as formatted string, 490–491, 493–494
  - time, setting, 494
  - tm structure, 492–493
  - Unix systems, 487–488
  - UTC (Coordinated Universal Time), 487
  - Windows systems, 488
  - Zulu time, 487
- debugging
  - error types *See also* error messages; errors
    - bugs, 42, 44–45
    - compiler errors, 42
    - endless loops, 514
    - linker errors, 42, 44, 504, 512
    - for loops, 514
    - mismatched parentheses, 513
    - parse errors, 43
    - run-time errors, 42, 44–45
    - sample program, 43–44
    - switch-case with no break, 513
    - syntax errors, 43
  - tools and techniques
    - ar tool, 515
    - egrep utility, 515–516
    - error-checking programs, 517
    - finding file contents, 515–516
    - formatting source code, 516–517
    - gdb (GNU Debugger), 515
    - grep utility, 515–516
    - indent utility, 516–517
    - ld (GNU linker) program, 517
    - lint, 517
    - make utility, 517–519
    - managing source code, 517–519
    - MinGW compiler, 515
    - synchronizing files, 519
    - timestamping files, 519
    - tool archives, 515
    - touch program, 519
- decimal (base 10) numbering system, 150
- decimal (floating-point) numbers. *See also* floating-point variables
  - absolute values, 179–180
  - from ASCII values, 74–75
  - as currency, 160–162
  - %E, conversion character, 159
  - %f, conversion character, 159

## 780 C All-in-One Desk Reference For Dummies

decimal (floating-point) numbers

(continued)

formatting, 160–162

%G, conversion character, 159

overview, 49–52

random numbers, 190

scientific (E) notation, 159–160

decimal points, aligning on, 161–162

decision making. *See also* goto statement

?: (question mark colon), ternary operator, 212–213

conditional operator, 212–213

else if statement, 84–87, 205–206

else statement, 83–84

falling through, 209–210

if statement

character comparisons, 80–82

equality comparisons, 80–81

greater-than comparisons, 81–82

less-than comparisons, 81–82

overview, 79–80

logical comparisons

== (equal signs), comparison operator, 130

AND operator, 127, 129–130

OR operator, 127–128

order of precedence, 132

Yes-or-No puzzle, 123–127, 131

multiple decisions, 84–87, 205–209

order of operations, 85–87

switch-case statement, 205–209

switch-case while(!done)

technique, 211–212

declaring

char variable, 64

constants, 104–105

functions, 14, 17, 294–295

structures, 281–283

variable type, 14, 17

variables, 52–53

variables, redeclaring, 146–148

decrementing values

- (minus sign), subtraction, 110

-- (minus signs), decrement operator, 115, 121

by one, 115–119

#define directive, 104–106

degrees *versus* radians, 171–173

deleteAccount() (delete account) function, 636–641

deleting

directories, 596

files, 610

items from double-linked linked lists, 661–666

records from linked lists, 635–641

dice. *See* craps game; rolling dice

difftime() (time difference) function, 494–496

dir() (directory) function, 600–602

directories (folders). *See also* files, disk closing, 588–590

creating, 596

current, shortcut, 602

deleting, 596

directory files, 591–593

finding (navigating), 593–597, 602–603, 672–674

location, changing, 595–597

location, determining, 593–595

navigating, 593–597, 602–603, 672–674

opening, 588–590

parent, shortcut, 602

reading files from, 590–591

regular files, 591–593

stepping through, 600–605

directory files, 591–593

directory pointers, 590

- disk files. *See also* directories;
  - header files; library files;
  - object code; source code
- binary *versus* text, 554–559
- buffers, 556
- closing, 550–551
- copying, 611
- creating, 548
- databases, 582–584
- definition, 524–526
- deleting, 610
- dumping contents of, 556–559
- finding contents of, 515–516
- formatted I/O, 562–563
- including, 17, 26–27
- managing, 670–672
- moving, 612–614
- opening, 547–550
- preventing overwrite, 552–554
- reading
  - arrays, 564–566
  - formatted, 563
  - random access, 576–582
  - structures, 570–571
  - unformatted, 551–552
- regular *versus* directory, 591–593
- renaming, 607–609
- size, viewing, 30
- synchronizing, 519
- timestamping, 519
- viewing, 555–556
- writing
  - arrays, 564
  - formatted, 562
  - random access, 581–582
  - structures, 569–570
  - unformatted, 547–550
- displaying. *See also* input/output;
  - printf() function; printing;
  - putchar() function;
  - puts() function; writing
- binary values
  - in binary, 395–398
  - in hexadecimal, 385–387
- files, 555–556
- on standard output device, 68
- strings, on screen, 244–245
- division
  - % (percent sign), modulo operator, 185–186
  - / (slash), division operator, 110
  - floating-point, 51
  - remainders, 185–186
- documentation. *See* books and publications; help; online resources
- dollar sign (\$), representing hexadecimal numbers, 154
- dollar sign question mark (\$?), Unix exit status, 545–546
- dot (.), current directory shortcut, 602
- dot dot (..), parent directory shortcut, 602
- double quotes (“)
  - in C statement syntax, 16
  - escape sequence for, 41
  - as literals in C statements, 41
- double variable
  - definition, 14
  - very large numbers, 140–141
- double-linked linked lists. *See also* linked lists
  - creating, 657–660
  - deleting items from, 661–666
  - for loops, 661
  - overview, 655–656
  - sample program, 656–661
  - scanning, 660–661
- do-while loops, 199–200
- dummy functions, 13
- dumping contents of files, 556–559

## 782 C All-in-One Desk Reference For Dummies

duplicating  
 files, 611  
 strings, 251–252  
 structure elements, 287–289

### E

%E, conversion character  
 definition, 771  
 floating-point numbers, 159  
 scientific (E) notation, 158

%e, conversion character  
 definition, 771  
 scientific (E) notation, 159

E (scientific) notation  
 %E, conversion character, 158  
 %e, conversion character, 159  
 %f, conversion character, 158  
 %G, conversion character, 158  
 %g, conversion character, 159  
 overview, 157–160

Easy Editor, 674

editors  
 case sensitivity, 25  
 Easy Editor, 674  
 MS-DOS Editor, 674  
 overview, 24–25  
 vim, 63, 674

egrep utility, 515–516

else if statement, 84–87, 205–206

else statement, 83–84

encryption, 530–532

END command, 13

ending programs  
 abnormally, 322–323  
 abort() function, 322–323  
 atexit() function, 325–327  
 exit() function, 322–323  
 normally, 322–323, 324–327  
 return codes, 319–322  
 return keyword, 319–322  
 shutdown routines, 324–327

endless loops. *See also* loops  
 debugging, 514  
 overview, 93–95  
 while loops, 201–202

ENIAC, 10

Enter key, equivalent in code. *See* \n,  
 newline character

entry keyword, 335

enum keyword, 336–337

enumerating variables, 336–337

the epoch, 487–488

equal sign (=), assignment operator, 23, 54

equal signs (==), comparison operator

definition, 130

string comparisons, 247

errno variable, 608–609

error codes (return codes)

exit status, 542–546

functions, 298

main() function, 14

from main() function, 319–322

error messages. *See also* debugging

controlling display of, 45–46

fatal errors, 45–46

and the GCC compiler, 45–46

level of detail, 34

verbosity, 34

warning errors, 45–46

error-checking programs, 517. *See also*

debugging

errorlevels (return codes)

exit status, 542–546

functions, 298

main() function, 14

from main() function, 319–322

errors. *See also* debugging

compiler, 42

linker, 42, 44

parse, 43

run-time, 42, 44–45

syntax, 43

escape sequences  
 \ (backslash), 35–36  
 for double quotes, 41  
 hexadecimal numbers, 155–156  
 list of, 769  
 \n, newline character, 35–36  
 \o, octal escape, 157  
 octal numbers, 157  
 overview, 35–36  
 \r, carriage return, 138  
 for single quotes, 63  
 \t, tab character, 76  
 tab character, 76  
 \x, hexadecimal escape, 155  
 examples. *See* sample programs  
 exercises, answers to, 685–761  
 exit codes (return codes)  
   exit status, 542–546  
   functions, 298  
   main() function, 14  
   from main() function, 319–322  
 EXIT command, 13  
 exit() (exit) function, 322–323  
 exit status, returning, 542–546  
 exiting programs. *See* ending programs  
 exponentiation, 167–169  
 extern keyword, 301, 501–503

---

## F

%f, conversion character  
   aligning numbers on decimal points,  
     161–162  
   definition, 771  
   floating-point numbers, 159  
   formatting numbers as currency,  
     160–161  
   output value width, setting, 161–162  
   scientific (E) notation, 158  
 falling through decision statements,  
   209–210

fclose() (file close) function,  
   550–551, 588  
 fflush() (flush input stream) function,  
   71–72, 78  
 file extensions  
   .C, 25  
   .CC, 25  
   .CPP, 25  
   .H, 27  
   header files, 27  
   .O, 28  
   object code files, 28  
   source code files, 25  
 file handles, 550  
 file pointers  
   definition, 578  
   position, determining, 578–579  
   position, resetting, 582  
   position, setting, 579–581  
 filenames, case sensitivity, 549  
 files, disk. *See also* directories;  
   header files; library files;  
   object code files; source code files  
   binary *versus* text, 554–559  
   buffers, 556  
   closing, 550–551  
   copying, 611  
   creating, 548  
   databases, 582–584  
   definition, 524–526  
   deleting, 610  
   dumping contents of, 556–559  
   finding contents of, 515–516  
   formatted I/O, 562–563  
   including, 17, 26–27  
   managing, 670–672  
   moving, 612–614  
   opening, 547–550  
   preventing overwrite, 552–554

## 784 C All-in-One Desk Reference For Dummies

- files, disk (*continued*)
  - reading
    - arrays, 564–566
    - formatted, 563
    - random access, 576–582
    - structures, 570–571
    - unformatted, 551–552
  - regular *versus* directory, 591–593
  - renaming, 607–609
  - size, viewing, 30
  - synchronizing, 519
  - timestamping, 519
  - viewing, 555–556
  - writing
    - arrays, 564
    - formatted, 562
    - random access, 581–582
    - structures, 569–570
    - unformatted, 547–550
- filters, input/output
  - encryption, 530–532
  - pig Latin, 532–536
  - rot13, 530–532
- finding
  - characters in pointer arrays, 436–439
  - file contents, 515–516
  - header files, 27
  - library files, 31
  - pointer arrays, 437–439
- flipping a coin, 190–192
- float variable
  - definition, 14, 137
  - storing and displaying characters, 67
  - using, 56
  - very large/small values, 139–140
- floating-point numbers. *See also* integers
  - absolute values, 179–180
  - from ASCII values, 74–75
  - as currency, 160–162
  - %E, conversion character, 159
  - %f, conversion character, 159
  - formatting, 160–162
  - %G, conversion character, 159
  - overview, 49–52
  - random numbers, 190
  - scientific (E) notation, 159–160
- floating-point variables
  - definition, 14, 137
  - storing and displaying characters, 67
  - using, 56
  - very large/small values, 139–140
- flow control (decision making). *See also* goto statement
  - ?: (question mark colon), ternary operator, 212–213
  - conditional operator, 212–213
  - else if statement, 84–87, 205–206
  - else statement, 83–84
  - falling through, 209–210
  - if statement
    - character comparisons, 80–82
    - equality comparisons, 80–81
    - greater-than comparisons, 81–82
    - less-than comparisons, 81–82
    - overview, 79–80
  - logical comparisons
    - == (equal signs), comparison operator, 130
    - AND operator, 127, 129–130
    - OR operator, 127–128
    - order of precedence, 132
    - Yes-or-No puzzle, 123–127, 131
  - multiple decisions, 84–87, 205–209
  - order of operations, 85–87
  - switch-case statement, 205–209
  - switch-case while(!done) technique, 211–212
- flushing the input stream
  - fflush() function, 71–72, 78
  - fpurge() (purge input stream) function, 71–72, 78
  - standard I/O, 527

- folders (directories). *See also* files, disk
  - closing, 588–590
  - creating, 596
  - current, shortcut, 602
  - deleting, 596
  - directory files, 591–593
  - finding (navigating), 593–597, 602–603, 672–674
  - location, changing, 595–597
  - location, determining, 593–595
  - navigating, 593–597, 602–603, 672–674
  - opening, 588–590
  - parent, shortcut, 602
  - reading files from, 590–591
  - regular files, 591–593
  - stepping through, 600–605
- fopen() (file open) function, 547–550, 588
- for loops. *See also* endless loops
  - counting by twos, 93
  - counting to 10, 91–92
  - debugging, 514
  - double-linked linked lists, 661
  - exit condition, 91
  - history of, 9
  - incrementing, 91
  - multiple conditions, 99–100
  - overview, 89–90
  - syntax, 90–91
  - using, 90–91
- formatted I/O, 562–563
- formatting. *See also* printf() function; scanf() function
  - floating-point numbers, 160–162
  - numbers as currency, 160–161
  - reading disk data, 563
  - reading disk files, 563
  - source code, 15–16, 516–517
  - time elements, 490–491, 493–494
  - time values, 490–491, 493–494
  - writing disk files, 562
- FORTTRAN, 10
- fpurge() (purge input stream) function, 71–72, 78
- fraction (floating-point) numbers. *See also* floating-point variables
  - absolute values, 179–180
  - from ASCII values, 74–75
  - as currency, 160–162
  - %E, conversion character, 159
  - %f, conversion character, 159
  - formatting, 160–162
  - %G, conversion character, 159
  - overview, 49–52
  - random numbers, 190
  - scientific (E) notation, 159–160
- fractional number variables, 14
- fread() (file read) function
  - definition, 566
  - disk-based databases, 582–584
  - reading directory contents, 588
  - reading selected records, 581
  - reading structures from disk, 571
- free() (free memory) function, 478–480
- freeing RAM, 478–480
- fseek() (file seek) function, 579–581, 582–584
- ftell() (file tell) function, 578–579
- functions. *See also* macros
  - \_ (underscores) in, 295
  - abort() (abort program), 322–323
  - atexit() (at exit), 325–327
  - atof() (ASCII-to-float conversion), 74–75, 78
  - atoi() (ASCII-to-integer conversion), 74–75, 78
  - BinString() (binary string), 395–397
  - calling, 295–296
  - chdir() (change directory), 595–597
  - closedir() (close directory), 589–590
  - ctime() (current time), 490–492
  - CTYPE.H character change, 266
  - CTYPE.H character tests, 266
  - declaring, 14, 17, 294–295

## 786 C All-in-One Desk Reference For Dummies

### functions (continued)

- deleteAccount() (delete account), 636–641
- difftime() (time difference), 494–496
- dir() (directory), 600–602
- dummy, 13
- ending programs, 322–327
- exit() (exit), 322–323
- extern keyword, 301
- fclose() (file close), 550–551, 588
- fflush() (flush input stream), 71–72, 78
- fopen() (file open), 547–550, 588
- fpurge() (purge input stream), 71–72, 78
- fread() (file read)
  - definition, 566
  - disk-based databases, 582–584
  - reading directory contents, 588
  - reading selected records, 581
  - reading structures from disk, 571
- free() (free memory), 478–480
- fseek() (file seek), 579–581, 582–584
- ftell() (file tell), 578–579
- fwrite() (file write)
  - databases to disk, 582–584
  - definition, 566
  - linked lists to disk, 644–645
  - random access files, 575–577
  - structures to disk, 569–570
- getchar() (get single characters). *See also* gets()
  - definition, 77
  - input stream, clearing, 71–72, 78
  - input stream, creating, 69–70
  - reading from the keyboard, 68–69
- getcwd() (get current working directory), 594–595
- gets() (get character strings). *See also* getchar()
  - bounds checking, 73
  - definition, 77
  - reading text streams, 72–73
  - security flaw, 73
- getting values from, 14–15
- help, 22, 422–424
- history of, 9
- input/output
  - arguments, 309
  - arrays as input, 310
  - both, 315–316
  - none, 297–298
  - returning values, 311–315
  - strings as input, 310
  - structures as input, 308–310
  - values as input, 304–308
- isalnum() (is alphanumeric), 266
- isalpha() (is alphabetical), 266–268
- isascii() (is ASCII), 266
- isblank() (is blank or tab), 266
- iscntrl() (is a control character), 266
- isdigit() (is a numeric digit), 266
- isgraph() (is a printable character), 266
- ishexnumber() (is a hex digit), 266
- islower() (is lowercase), 266
- isprint() (is a printable character), 266
- ispunct() (is punctuation), 266
- isspace() (is white space), 266–268
- isupper() (is uppercase), 266
- isxdigit() (is a hex digit), 266
- main()
  - adding actions to, 16
  - argc argument, 538–539
  - arguments, 13
  - argv argument, 538, 539–540
  - calling, 13
  - counting arguments, 539
  - declaring type, 14
  - getting values from, 14–15
  - overview, 12–13
  - return codes, 14
  - syntax, 13
  - viewing arguments, 539–540

- malloc() (memory allocation). *See also* realloc() (reallocate memory)
  - creating structures, 480–484
  - overview, 472–477
- math. *See also* mathematical operations
  - abs() (absolute values), 177–180
  - cbrt() (cube root), 169–170
  - cos() (cosine of an angle), 173–177
  - exponentiation, 167–169
  - linking to the math library, 167
  - log() (natural logarithms), 170
  - log10() (base 10 logarithms), 170
  - pow() (raising to a power), 167–169
  - radians *versus* degrees, 171–173
  - rand() (random number), 181
  - random() (random number), 182–185
  - sin() (sine of an angle), 173–177
  - sqrt() (square root), 166–167
  - srand() (seed random number), 183–185
  - srandom() (seed random number), 183–185
  - tan() (tangent of an angle), 173–177
  - trigonometry, 171–177
- mkdir() (make directory), 596
- names, uniqueness, 505
- naming, 295
- opendir() (open directory), 588–590
- overview, 21–23
- printf() (print)
  - aligning numbers on decimals, 161–162
  - definition, 77
  - displaying hexadecimal numbers, 153–154
  - displaying single characters, 64–65
  - formatting floating-point numbers, 160–162
  - formatting numbers as currency, 160–162
  - justifying text, 163–164
  - output value width, setting, 161–163
  - printing text, 35
- prototyping, 296–297, 316–317
- putchar() (put single characters)
  - definition, 77
  - displaying single characters, 64–65
- puts() (put character strings)
  - definition, 77
  - displaying to a screen, 16
- readdir() (read directory), 590–591
- realloc() (reallocate memory), 477–478
- rename() (rename file), 607–609
- retaining values after use, 337–341
- return codes, 298
- return keyword, 298, 311–315
- rewind() (rewind), 582
- rmdir() (remove directory), 596
- scanf() (format input)
  - getting a variable's address, 358
  - getting an array's address, 365–366
  - reading from the keyboard, 75–77
- self calling (recursion), 597–605
- shutdown routines, 324–327
- spaces in, 295
- square root calculation, 22
- strcasecmp() (string comparison), 251, 254
- strcat() (string concatenation), 251
- strchr() (string character location), 251
- strcmp() (string comparison), 251, 253–254
- strcpy() (string copy), 251, 252
- string processing, 250–251
- strings, 250–251
- strlen() (string length), 251, 256–257
- strncasecmp() (string comparison), 251
- strncat() (string concatenation), 251, 254–256
- strncpy() (string copy), 251

## 788 C All-in-One Desk Reference For Dummies

### functions (continued)

strchr() (string character location), 251  
 strstr() (string locate), 251  
 system() (system), 541–542  
 time() (time), 489–490  
 toascii() (convert to ASCII), 266  
 tolower() (convert to lowercase), 266, 269–270  
 toupper() (convert to uppercase), 266, 268–269  
 unlink() (unlink file), 610  
 variables  
   global, 300–303  
   local, 298–300  
 functions, and pointers  
   array notation, 461–462  
   arrays  
     modifying, 463–464  
     passing to functions, 460–464  
     returning from functions, 464–466  
     size, determining, 462  
   functions that accept pointers, 453–455  
   passing pointers to functions, 456–457  
   pointers *versus* values, 452–453  
   returning pointers from functions, 457–459  
   strings  
     passing to functions, 467–468  
     returning from functions, 468–469  
 fwrite() (file write) function  
   databases to disk, 582–584  
   definition, 566  
   linked lists to disk, 644–645  
   random access files, 575–577  
   structures to disk, 569–570

## G

%G, conversion character  
   definition, 771  
   floating-point numbers, 159  
   scientific (E) notation, 158

%g, conversion character  
   definition, 771  
   scientific (E) notation, 159  
 GCC compiler. *See also* compiler; linkers  
   -c option, 500  
   compiling multiple modules, 500  
   compiling without linking, 28, 500  
   displaying warnings, 45–46  
   linking math libraries, 31  
   linking to the math library, 167  
   -lm option, 31, 167  
   naming object files, 30, 501  
   -o option, 30, 501  
   online resources, 22  
   random numbers, 181, 183  
   setting up, 670  
   on Unix, 678–680  
   -Wall option, 45–46  
   on Windows, 675–677  
 gdb (GNU Debugger), 515  
 getchar() (get single characters)  
   function. *See also* gets()  
   definition, 77  
   input stream, clearing, 71–72, 78  
   input stream, creating, 69–70  
   reading from the keyboard, 68–69  
 getcwd() (get current working directory) function, 594–595  
 gets() (get character strings) function.  
   *See also* getchar()  
   bounds checking, 73  
   definition, 77  
   reading text streams, 72–73  
   security flaw, 73  
 getting. *See also* input/output; reading  
   array addresses, 365–366  
   array size, 353–355  
   time elements  
     current day, 491–492  
     as epoch value, 488–490  
     as formatted string, 490–491, 493–494  
   values from functions, 14–15  
   variable addresses, 355–358  
   variable size, 352–353

global variables  
 programs with multiple modules,  
 501–503  
 sample program, 301–303  
 GMT (Greenwich Mean Time), 487  
 GNU Debugger (gdb), 515  
 GNU linker (ld) program, 517  
 goto statement, 217–220. *See also*  
 decision making  
 Greenwich Mean Time (GMT), 487  
 Gregorian calendar, 486–487  
 grep utility, 515–516

---

## H

---

.H file extension, 27  
 header files  
 creating, 27, 504–505  
 definition, 17  
 file extension, 27  
 finding, 27  
 including in source code, 26–27  
 help. *See also* books and publications;  
 online resources  
 for functions, 22, 422–424  
 library documentation, 422–424  
 man command, 22  
 hexadecimal (base 16) numbering  
 system  
 case sensitivity, 154  
 displaying, 151, 163  
 displaying binary values, 385–387  
 escape sequences, 155–156  
 identifying, 151  
 overview, 151–153  
 representing, 151, 154, 163  
 Hopper, Grace, 10  
 Hungarian notation, 329–331

---

## I

---

%i, conversion character, 771  
 if errorlevel command, 544–545

if statement  
 character comparisons, 80–82  
 equality comparisons, 80–81  
 greater-than comparisons, 81–82  
 less-than comparisons, 81–82  
 overview, 79–80  
 IF-THEN statement, 9  
 immediate addition, 49  
 immediate values, 48  
 #include directive, 17, 26–27  
 include folder, 27  
 including files in source code, 17, 26–27  
 incrementing values  
 + (plus sign), addition operator, 110  
 ++ (plus signs), increment operator,  
 115, 121  
 immediate, 49  
 for loops, 91  
 by more than one, 119–120  
 by one, 115–119  
 pre/post-incrementing, 117–119  
 values and addresses, 408–410  
 indent utility, 516–517  
 infinite loops. *See* endless loops  
 initializing  
 arrays, 228–229, 236–239  
 pointers, 355, 358, 362–365  
 variables, 53–54  
 input stream  
 clearing, 71–72, 78  
 creating, 69–70  
 input/output. *See also* displaying;  
 printing; reading; writing  
 filters  
 encryption, 530–532  
 pig Latin, 532–536  
 rot13, 530–532  
 functions  
 arguments, 309  
 arrays as input, 310  
 both, 315–316  
 none, 297–298  
 returning values, 311–315  
 strings as input, 310

## 790 C All-in-One Desk Reference For Dummies

---

input/output, functions (*continued*)  
   structures as input, 308–310  
   values as input, 304–308  
 standard I/O  
   devices, 68, 525  
   flushing, 527  
   input device, 68  
   input redirection, 527  
   output device, 68  
   output piping, 528–529  
   output redirection, 528  
   overview, 524–526  
   requirements for, 523–524  
   sample program, 526–530  
   STDIO.H file, 525–526

int variable  
   definition, 14, 136  
   long, 67, 137–139  
   short, 67, 137–139  
   storing and displaying characters, 67  
   using, 54–55

integer variable  
   definition, 14, 136  
   long, 67, 137–139  
   short, 67, 137–139  
   storing and displaying characters, 67  
   using, 54–55

integers. *See also* floating-point numbers  
   from ASCII values, 74–75  
   in char variables, 65–67  
   definition, 48–49

Internet resources. *See* online resources

isalnum() (is alphanumeric) function,  
 266

isalpha() (is alphabetical) function,  
 266–268

isascii() (is ASCII) function, 266

isblank() (is blank or tab) function,  
 266

isctrl() (is a control character)  
 function, 266

isdigit() (is a numeric digit) function,  
 266

isgraph() (is a printable character)  
 function, 266

ishexnumber() (is a hex digit) function,  
 266

islower() (is lowercase) function, 266

isprint() (is a printable character)  
 function, 266

ispunct() (is punctuation) function,  
 266

isspace() (is white space) function,  
 266–268

isupper() (is uppercase) function, 266

isxdigit() (is a hex digit) function, 266

---

### J

jargon, programmer, 511

Java, 10

Julian dates, 486

---

### K

keyboard

  entering nonstandard characters, 35–36.  
   *See also* escape sequences

  input with conversion characters  
   (placeholders), 77

  reading from, 68–69

  reading strings, 244–245. *See also*

    getchar() function; gets()

    function; scanf() function

  standard input device, 68

keywords

  ISO C standard, 763

  list of, 21, 763–765

  overview, 20–21

  reserved, 763

  killing programs. *See* ending programs

---

## L

---

- ld (GNU linker) program, 517
- left angle bracket (<), input redirection, 527
- left angle brackets (<<), shift bits left, 388, 390–391
- Leviathan, 473
- lib directory, 31
- library documentation, 422–424
- library files
  - definition, 29–30
  - finding, 31
  - math, linking to, 31
- Lilith, 473
- linked lists. *See also* databases; structures
  - adding structures, 622–626
  - building structures, 620–622
  - double-linked
    - creating, 657–660
    - deleting items from, 661–666
    - for loops, 661
    - overview, 655–656
    - sample program, 656–661
    - scanning, 660–661
  - filling structures, 624–626
  - linking to structures, 622–624
  - marking the end, 626–628
  - nodes, 629
  - NULL pointer, 626–628
  - overview, 619–620
  - reading from disk, 645–649
  - records
    - definition, 629
    - deleting, 635–641
    - sample program, 630–641, 648–654
    - typedef keyword, 333–334, 626
    - write confirmation, 645
    - writing to disk, 644–645
- linkers. *See also* GCC compiler error checking, 42, 44, 504, 512
  - GNU linker (ld) program, 517
  - overview, 29–30
- linking. *See also* compiling
  - with compiling, 675–680
  - without compiling, 28, 500
  - GNU linker (ld) program, 517
  - to math libraries, 31, 167
  - to the math library, 167
  - object code, 29–30
  - program modules, 509–510
  - programs with multiple modules, 498–501
  - structures to linked lists, 622–624
- lint program, 517
  - lm option, 31
- local variables, 298–300
- locating
  - characters in pointer arrays, 436–439
  - file contents, 515–516
  - header files, 27
  - library files, 31
  - pointer arrays, 437–439
- location, RAM. *See* addresses
- log() (natural logarithms) function, 170
- log10() (base 10 logarithms) function, 170
- logical comparisons
  - == (equal signs), comparison operator, 130
  - AND operator, 127, 129–130
  - OR operator, 127–128
  - order of precedence, 132
  - Yes-or-No puzzle, 123–127, 131
- logical operations, 391–395
- logical operators
  - & (ampersand), AND, 391–394
  - ^ (caret), XOR (exclusive OR), 399–400
  - | (vertical bar), OR, 394–395, 397–398
- long int variable, 67

## 792 C All-in-One Desk Reference For Dummies

long words, definition, 385

looking for

- characters in pointer arrays, 436–439
- file contents, 515–516
- header files, 27
- library files, 31
- pointer arrays, 437–439

loops. *See also* recursion

- for
  - counting by twos, 93
  - counting to 10, 91–92
  - double-linked linked lists, 661
  - exit condition, 91
  - incrementing, 91
  - multiple conditions, 99–100
  - overview, 89–90
  - syntax, 90–91
  - using, 90–91
- break keyword, 96
- breaking out of, 96
- continue keyword, 202–203
- continuing, 202–203
- counting by letters, 98
- do-while, 199–200
- endless
  - debugging, 514
  - overview, 93–95
  - while loops, 201–202
- names of God, 98–99
- negative numbers, 141–142
- nesting, 97, 203–204
- parts of, 89–90
- while
  - endless, 201–202
  - nested, 203–204
  - overview, 193–197
  - while(!done), 197–199, 211–212

Lotto sample program

- creating a folder for, 503
- creating header files, 504–505
- creating modules, 504, 506–509

- linking the modules, 509–510
- picking lottery numbers, 240–241
- specifying modules in sequence, 506
- unique function names, 505

Lucifer, 473

## M

Macintosh compiler, 670

macros, 265–266. *See also* functions

main() function

- adding actions to, 16
- argc argument, 538–539
- arguments, 13
- argv argument, 538, 539–540
- calling, 13
- declaring type, 14
- ending
  - abnormally, 322–323
  - abort() (abort program) function, 322–323
  - atexit() (at exit) function, 325–327
  - exit() (exit) function, 322–323
  - normally, 322–323, 324–327
  - return codes, 319–322
  - return keyword, 319–322
  - shutdown routines, 324–327
- getting values from, 14–15
- overview, 12–13
- return codes, 14
- syntax, 13

make utility, 517–519

malloc() (memory allocation) function

- creating structures, 480–484
- overview, 472–477

man command, 22

Marduk, 473

math libraries, linking to, 31, 167

mathematical operations. *See also*

- functions, math; numbering systems;
- numbers; operators, math

- addition
  - + (plus sign), addition operator, 110
  - ++ (plus signs), increment operator, 115, 121
  - immediate, 49
  - incrementing by more than one, 119–120
  - incrementing by one, 115–119
  - incrementing values and addresses, 408–410
  - for loops, 91
  - pre/post-incrementing, 117–119
- decrementing values
  - (minus sign), subtraction, 110
  - (minus signs), decrement operator, 115, 121
  - by one, 115–119
- division
  - % (percent sign), modulo operator, 185–186
  - / (slash), division operator, 110
  - floating-point, 51
  - remainders, 185–186
- floating-point, 50
- incrementing values
  - + (plus sign), addition operator, 110
  - ++ (plus signs), increment operator, 115, 121
  - immediate, 49
  - for loops, 91
  - by more than one, 119–120
  - by one, 115–119
  - pre/post-incrementing, 117–119
  - values and addresses, 408–410
- multiplication
  - \* (asterisk), multiplication operator, 110
  - floating-point, 50
- negating a value, 114–115
- pointers, 367–371
- random numbers. *See also* random()
  - function
    - craps game, 213–216, 312–315
    - flipping a coin, 190–192
    - floating-point numbers, 190
    - GCC compiler, 181, 183
    - pseudorandom, 181
    - rolling dice, 187–190, 213–216, 312–315
    - seed numbers, 183–185
    - trimming, 186–187
  - subtraction
    - (minus sign), subtraction, 110
    - (minus signs), decrement operator, 115, 121
    - decrementing by one, 115–119
    - variables, on themselves, 120–121
- memory. *See also* addresses (storage location)
  - allocating. *See also* malloc() function; realloc() function
    - arrays, 480–484
    - structures, 480–484
    - variables, 351–352
  - contents of, 350–351
  - definition, 349–350
  - freeing, 478–480
  - previewing contents, 374–376
  - saving space, 431–432
  - size, arrays, 353–355
  - size, variables, 352–353
  - sizeof operator, 353
  - variable addresses, 355–358
- MinGW compiler, 515, 670
- minus equal (=), shortcut operator, 121
- minus sign (-)
  - subtraction, 110
  - unary negative operator, 110
- minus signs (--), decrement operator, 115, 121
- mismatched parentheses, debugging, 513

## 794 C All-in-One Desk Reference For Dummies

- mkdir() (make directory) function, 596
  - Modified Julian Day, 486
  - modules
    - creating, 504, 506–509
    - linking, 509–510
  - Molech, 473
  - Moloch, 473
  - monitor. *See* displaying
  - move (mv) command, 612–614
  - moving files, 612–614
  - MS-DOS Editor, 674
  - multidimensional arrays
    - 2 dimensions, 236–239, 236–239
    - 3 dimensions, 239–240, 259–261
    - structure of, 236
  - multiple variable, 56–57
  - multiplication
    - \* (asterisk), multiplication operator, 110
    - floating-point, 50
  - mv (move) command, 612–614
- 
- ### N
- 
- \n, newline character, 35–36
  - names
    - files
      - case sensitivity, 549
      - renaming, 607–609
    - functions, 295, 505
    - of God, 98–99
    - object files, 30, 501
    - pointers, 360
    - variables
      - all caps, 332
      - Hungarian notation, 329–331
      - naming conventions, 329–331
      - prefixes, 330
      - renaming, 331–332. *See also* typedef keyword
      - uniqueness, 306
  - navigating directories, 593–597, 602–603
  - negating a value, 114–115
  - negative numbers
    - in loops, 141–142
    - square root of, 167
    - in variables, 141–146
  - nesting
    - comments, 39–40
    - loops, 97, 203–204
    - structures, 289–291
  - Neumann, John von, 9
  - The New Hacker's Dictionary*, 511
  - new keyword, 623
  - The Nine Billion Names of God*, 98–99
  - nodes, 629
  - nonstandard characters. *See also* escape sequences; operators
    - hexadecimal representation, 155–156
    - using in programs, 35–36
  - NULL
    - ending linked lists, 626–628
    - ending strings, 248–250
  - numbering systems
    - base 2 (binary). *See also* binary operations
      - counting by twos, 382–384
      - definition, 151
      - displaying binary values, 385–387, 395–398
    - base 8 (octal), 156–157
    - base 10 (decimal), 150
    - base 16 (hexadecimal)
      - case sensitivity, 154
      - displaying, 151, 163
      - escape sequences, 155–156
      - identifying, 151
      - overview, 151–153
      - representing, 151, 154, 163
    - counting bases, 150
    - scientific (E) notation, 157–160
  - numbers. *See also* floating-point numbers; integers; mathematical operations; operators
    - aligning on decimal points, 161–162
    - commas in source code, 56

conversion characters  
 (placeholders), 48  
 formatting as currency, 160–162  
 immediate addition, 49  
 immediate values, 48  
 negative  
 in loops, 141–142  
 square root of, 167  
 in variables, 141–146  
 output value width, setting, 161–163

## O

%o, conversion character  
 definition, 771  
 displaying hexadecimal numbers, 163  
 sample program, 156–157  
 \o, octal escape, 157  
 .O file extension, 28  
 -o option, 30, 501  
 object code files. *See also* source  
 code files  
 compiling without linking, 28  
 definition, 25  
 file extension, 28  
 managing, 670–672  
 naming, 30  
 size, viewing, 30  
 storing, 670–672  
 object-oriented programming (OOP), 10  
 octal (base 8) numbering system,  
 156–157  
 one dimensional arrays  
 creating, 224–225  
 declaring empty, 226–228  
 definition, 223  
 refilling, 228–229  
 resizing, 227  
 sorting, 229–235  
 using, 225–226

one's complement, 400–402  
 online resources  
 Apple, 670  
*C All-in-One Desk Reference For  
 Dummies*, 3, 670  
*C For Dummies*, 612  
 Code Warrior compiler, 670  
 GCC compiler, 22  
 gdb (GNU Debugger), 515  
 help for functions, 22  
 Macintosh compilers, 670  
 MinGW compiler, 670  
 source code files, 4  
 Windows compilers, 670  
 OOP (object-oriented programming), 10  
 opendir() (open directory) function,  
 588–590  
 opening. *See also* reading  
 directories, 588–590  
 files, 547–550  
 operators  
 assignment, 764–765  
 bitwise  
 << (left angle brackets), shift left, 388,  
 390–391  
 >> (right angle brackets), shift right,  
 388–390  
 list of, 387  
 C language, 763–764  
 logical  
 AND, 127, 129–130  
 == (equal signs), comparison  
 operator, 130  
 OR, 127–128  
 order of precedence, 132  
 logical binary  
 & (ampersand), AND, 391–394  
 ^ (caret), XOR (exclusive OR), 399–400  
 | (vertical bar), OR, 394–395, 397–398  
 order of precedence, 110–114

## 796 C All-in-One Desk Reference For Dummies

operators, math. *See also* nonstandard characters

- \* (asterisk), multiplication operator
  - definition, 110
  - floating-point multiplication, 50
- \*= (asterisk equal), shortcut operator, 121
- (minus equal), shortcut operator, 121
- (minus sign), subtraction, 110
- (minus sign), unary minus, 114–115
- (minus signs), decrement operator, 115, 121
- += (plus equal), shortcut operator, 121
- + (plus sign), addition, 110
- + (plus sign), unary plus, 114–115
- ++ (plus signs), increment operator, 115, 121
- / (slash), division operator
  - definition, 110
  - floating-point division, 51
- /= (slash equal), shortcut operator, 121
- assignment, 764–765
- list of, 763–765
- order of precedence, 765–766
- unary operators, 114–115

operators, unary

- & (ampersand), address operator
  - initializing pointers, 355, 358, 365
  - in `scanf()` function, 358, 365
- \* (asterisk), pointer operator
  - initializing pointers, 364
  - modifying variables, 376–380
  - previewing RAM contents, 374–376
  - requirements for, 361
  - syntax, 360
- (minus sign), negative operator, 114–115
- + (plus sign), positive operator, 114–115
- ~ (tilde), one's complement, 400–402
- overview, 114–115
- `sizeof` (return size of object), 353

- OR operator, 127–128
- order of precedence
  - % (percent sign), modulo operator, 186
  - logical operators, 132
  - math operators, 110–114
  - operators, 765
  - variables, 144
- output. *See* displaying; input/output; printing; writing
- overflow, variables, 138–139

### p

- %p, conversion character, 771
- pagan deities, 473
- papers. *See* books and publications
- parameters. *See* arguments
- parent directories, shortcut, 602
- parentheses (())
  - function syntax, 13
  - mismatched, debugging, 513
- parse errors, 43
- Pascal, 10
- percent sign (%), modulo operator, 185–186
- percent signs (%%), conversion character, 771
- pig Latin filter, 532–536
- placeholders (conversion characters), 771. *See also specific placeholders*
  - integers, 48
  - and keyboard input, 77. *See also* `scanf()` function
  - list of, 771
- plus equal (+=), shortcut operator, 121
- plus sign (+)
  - addition operator, 110
  - unary positive operator, 110
- plus signs (++), increment operator, 115, 121

- pointer arrays
  - \*\* (asterisks), pointer-pointer prefix, 443
  - array notation, 437–439, 441–443
  - comparing strings, 446–447
  - creating, 432–434
  - finding, 437–439
  - finding characters in, 436–439
  - to other pointers, 439–441
  - overview, 429–431
  - sample program, 434–436
  - saving space, 431–432
  - sorting strings, 443–449
- pointer variables, table of, 410
- pointers. *See also* RAM
  - to array addresses, 364–366
  - creating, 360–361
  - definition, 359
  - directory, 590
  - file, 578–582
  - initializing, 362–364
  - math, 367–371
  - names, 360
  - passing to functions, 456–457
  - replacing arrays
    - [ ] (square brackets), array brackets, 411–413
    - array notation, eliminating, 405, 406–408, 411–415
    - filling arrays with pointers, 405–406
    - incrementing values and addresses, 408–410
    - pointer variables, table of, 410
    - returning from functions, 457–459
    - sample program, 359–360, 362–364
    - sharing, 378–380
  - and strings
    - declaring strings, 426–427
    - displaying strings, 417–422
    - strings *versus* characters, 422–426
    - versus* values, 452–453
    - to variable addresses, 360–364
- pointers, and functions
  - array notation, 461–462
  - arrays
    - modifying, 463–464
    - passing to functions, 460–464
    - returning from functions, 464–466
    - size, determining, 462
  - functions that accept pointers, 453–455
  - passing pointers to functions, 456–457
  - pointers *versus* values, 452–453
  - returning pointers from functions, 457–459
  - strings
    - passing to functions, 467–468
    - returning from functions, 468–469
- post-incrementing, 117–119
- pound sign (#), precompiler directives, 25
- pow() (raising to a power) function, 167–169
- precision, variables, 58, 140–141
- pre-incrementing, 117–119
- prime numbers, 405
- printf() (print) function
  - aligning numbers on decimals, 161–162
  - definition, 77
  - displaying hexadecimal numbers, 153–154
  - displaying single characters, 64–65
  - formatting floating-point numbers, 160–162
  - formatting numbers as currency, 160–162
  - justifying text, 163–164
  - output value width, setting, 161–163
  - printing text, 35
- printing. *See also* displaying
  - aligning numbers on decimals, 161–162
  - definition, 77
  - displaying hexadecimal numbers, 153–154

## 798 C All-in-One Desk Reference For Dummies

---

### printing (*continued*)

- displaying single characters, 64–65
- formatting floating-point numbers, 160–162
- formatting numbers as currency, 160–162
- justifying text, 163–164
- output value width, setting, 161–163
- printing text, 35

problem solving. *See* debugging

procedures. *See* functions

program skeleton. *See also* keywords;  
`main()` function; operators

- basic program, 11–12

- example, 18

- exit, 13–14

programmer jargon, 511

programming languages, 20. *See also*  
 sample programs; *specific languages*

programs. *See also* object code files;  
 source code files

- decision making *See also* goto  
 statement

- ?: (question mark colon), ternary  
 operator, 212–213

- conditional operator, 212–213

- else if statement, 84–87, 205–206

- else statement, 83–84

- falling through, 209–210

- if statement, 79–82

- logical comparisons, 123–132

- multiple decisions, 84–87, 205–209

- order of operations, 85–87

- switch-case statement, 205–209

- switch-case while(!done)

- technique, 211–212

including files in, 17, 26–27

text in, 16

programs, creating

- See* compilers

- See* compiling

- See* editors

- See* linking

programs, with multiple modules

- command prompt, 498

- compiling, 498–501

- extern keyword, 501–503

- global variables, 501–503

- linking, 498–501

- Lotto program, sample

- creating a folder for, 503

- creating header files, 504–505

- creating modules, 504, 506–509

- linking the modules, 509–510

- specifying modules in sequence, 506

- unique function names, 505

- sharing variables, 501–503

prototyping functions, 296–297, 316–317

pseudorandom numbers, 181

publications. *See* books and publications

purging the input stream

- `fflush()` function, 71–72, 78

- `fpurge()` (purge input stream)

- function, 71–72, 78

- standard I/O, 527

`putchar()` (put single characters)

- function

- definition, 77

- displaying single characters, 64–65

`puts()` (put character strings) function

- definition, 77

- displaying to a screen, 16

---

## Q

question mark colon (:), ternary  
 operator, 212–213

quitting programs. *See* ending programs

---

## R

`\r`, carriage return, 138

radians *versus* degrees, 171–173

- RAM (random access memory). *See also*  
 addresses (storage location)  
 allocating *See also* `malloc()` function;  
   `realloc()` function  
 arrays, 480–484  
 structures, 480–484  
 variables, 351–352  
 contents of, 350–351  
 definition, 349–350  
 freeing, 478–480  
 previewing contents, 374–376  
 saving space, 431–432  
 size, arrays, 353–355  
 size, variables, 352–353  
`sizeof` operator, 353  
 variable addresses, 355–358
- `rand()` (random number) function, 181
- random access memory (RAM). *See also*  
 addresses (storage location)  
 allocating *See also* `malloc()` function;  
   `realloc()` function  
 arrays, 480–484  
 structures, 480–484  
 variables, 351–352  
 contents of, 350–351  
 definition, 349–350  
 freeing, 478–480  
 previewing contents, 374–376  
 saving space, 431–432  
 size, arrays, 353–355  
 size, variables, 352–353  
`sizeof` operator, 353  
 variable addresses, 355–358
- `random()` (random number) function,  
 182–185
- random numbers  
 craps game, 213–216, 312–315  
 flipping a coin, 190–192  
 floating-point numbers, 190  
 GCC compiler, 181, 183  
 lottery numbers, 240–241  
 pseudorandom, 181  
 rolling dice, 187–190, 213–216, 312–315  
 sample programs, 213–216  
 seed numbers, 183–185  
 trimming, 186–187
- Raymond, Eric, 511
- readability, source code, 15–16
- `readdir()` (read directory) function,  
 590–591
- reading. *See also* `getchar()` function;  
   `gets()` function; input/output;  
   opening; `scanf()` function  
 arrays from disk, 564–566  
 directory contents, 588  
 disk data, 563  
 files  
   from directories, 590–591  
   formatted, 563  
   random access, 576–582  
   selected records, 581  
   unformatted, 551–552  
 linked lists from disk, 645–649  
 strings, 244–245  
 structures, 566–574
- `realloc()` (reallocate memory)  
 function, 477–478
- records  
 linked lists  
   definition, 629  
   deleting, 635–641  
   structures, 273–274
- recursion, 597–605. *See also* loops
- register keyword, 337
- registers, placing variables in, 337
- regular files, 591–593
- remainders, 185–186
- `rename()` (rename file) function, 607–609
- renaming  
 files, 607–609  
 variables, 331–332. *See also* `typedef`  
 keyword

## 800 C All-in-One Desk Reference For Dummies

repeating statements. *See* loops  
 resizing arrays, 227  
 return codes (error codes)  
   exit status, 542–546  
   functions, 298  
   main() function, 14  
   from main() function, 319–322  
 return keyword  
   main() function, 319–322  
   omitting, 298  
   requirements for, 319–322  
   returning values, 311–315  
   syntax, 14–15  
 returning values from functions, 311–315  
 rewind() (rewind) function, 582  
 right angle bracket (>), output  
   redirection, 528  
 right angle brackets (>>), shift bits right,  
 388–390  
 Ritchie, Dennis, 10  
 rmdir() (remove directory) function,  
 596  
 rolling dice, 187–190, 213–216, 312–315.  
   *See also* craps game  
 rot13 filter, 530–532  
 routines. *See* functions; macros  
 run-time errors, 42, 44–45

## S

%s, conversion character, 771  
 sample programs  
   banking, 630–641, 648–654  
   constants, 101–103, 105  
   craps game, 213–216, 312–315  
   databases, 630–641, 648–654  
   debugging, 43–44  
   double-linked linked lists, 656–661  
   exercises, answers to, 685–761  
   flipping a coin, 190–192  
   global variables, 301–303  
   “hello” example, 33–34

linked lists, 630–641, 648–654  
 Lotto  
   creating a folder for, 503  
   creating header files, 504–505  
   creating modules, 504, 506–509  
   linking the modules, 509–510  
   picking lottery numbers, 240–241  
   specifying modules in sequence, 506  
   unique function names, 505  
 metric conversion, 359–360, 362–364  
 %o, conversion character, 156–157  
 pointer arrays, 434–436  
 pointers, 359–360, 362–364  
 random numbers, 187–192, 213–216,  
 240–241  
 reading and writing structures, 566–574  
 rolling dice, 187–189, 213–216, 312–315  
 standard I/O, 526–530  
 STOP program, 34  
 %u, conversion character, 145–146  
 variables, 137  
 Yes-or-No puzzle, 123–127, 131  
 scanf() (format input) function  
   getting a variable’s address, 358  
   getting an array’s address, 365–366  
   reading from keyboard, 75–77  
 scanning double-linked linked lists,  
 660–661  
 scientific (E) notation  
   %E, conversion character, 158  
   %e, conversion character, 159  
   %f, conversion character, 158  
   %G, conversion character, 158  
   %g, conversion character, 159  
   overview, 157–160  
 searching for  
   characters in pointer arrays, 436–439  
   file contents, 515–516  
   header files, 27  
   library files, 31  
   pointer arrays, 437–439

- security, gets() flaw, 73
- seed numbers, 183–185
- semicolon (;)
  - C statements, 15–16, 24
  - constants, 104
  - function input, 304
  - precompiler directives, 25
- sharing variables, 501–503
- shifting bits, 388–391
- Short Code, 10
- short int variable, 67
- shutdown routines, 324–327
- signed variables, 142–143, 145
- Simonyi, Charles, 329
- sin() (sine of an angle) function, 173–177
- single quote (')
  - in character variables, 62
  - escape sequences, 63
  - as literal character, 63
- sizeof operator, 353
- slash (/), division operator
  - definition, 110
  - floating-point division, 51
- slash asterisk... (/\*...\*/), comment delimiters, 36–37
- slash equal (/=), shortcut operator, 121
- slashes (/), comment delimiter, 40
- software machines. *See* functions
- sorting
  - arrays, 229–235
  - strings, 443–449
- source code files. *See also* object
  - code files; programs
  - commas in numbers, 56
  - comments
    - creating, 36–37, 40
    - disabling code with, 38
    - nesting, 39–40
  - definition, 20
  - file extension, 25
  - formatting, 15–16, 516–517
  - including files in, 17, 26–27
  - line breaks, 35–36
  - managing, 517–519, 670–672
  - parsing, 28
  - readability, 15–16
  - reediting, 34
  - starting a new line, 35–36
  - storing, 670–672
- spaces, in functions, 295
- special characters. *See* nonstandard characters
- sqrt() (square root) function, 166–167
- square brackets ([ ]), array brackets, 411–413
- srand() (seed random number) function, 183–185
- srandom() (seed random number) function, 183–185
- standard I/O. *See also* input/output devices, 525
  - flushing, 527
  - input redirection, 527
  - output piping, 528–529
  - output redirection, 528
  - overview, 524–526
  - requirements for, 523–524
  - sample program, 526–530
  - STDIO.H file, 525–526
- standard output device, 68
- static keyword, 337–340
- STDIO.H file, 525–526
- stopping programs. *See* ending programs
- Stoustroup, Bjarne, 10, 117
- strcasecmp() (string comparison) function, 251, 254
- strcat() (string concatenation) function, 251
- strchr() (string character location) function, 251
- strcmp() (string comparison) function, 251, 253–254

## 802 C All-in-One Desk Reference For Dummies

- strcpy() (string copy) function, 251, 252
- string processing function, 250–251
- strings. *See also* text
  - case, converting to lower, 266, 269–270
  - case, converting to upper, 266, 268–269
  - case sensitivity, 251, 254
  - changing, functions for, 266
  - versus* characters, 422–426
  - comparing, 247, 253–254, 446–447
  - concatenating, 254–256
  - copying, 251–252
  - declaring, 426–427
  - definition, 23
  - displaying, 417–422
  - functions, 250–251, 266
  - input to functions, 310
  - joining (concatenating), 254–256
  - multicharacter
    - \0, NULL character, 248–250
    - as arrays, 247
    - arrays of, 257–263
    - assigning to other strings, 246–247
    - counting characters, 267–268
    - displaying on screen, 244–245. *See also* printf() function; puts() function
    - length, determining, 256–257
    - processing character by character, 248–250
    - reading, 244–245. *See also* gets() function; scanf() function
    - structure of, 247
    - terminating, 248
    - terminator character, 248–250
    - writing, 244–245. *See also* printf() function; puts() function
  - passing to functions, 467–468
  - and pointers
    - declaring strings, 426–427
    - displaying strings, 417–422
    - strings *versus* characters, 422–426
    - returning from functions, 468–469
  - single character. *See also* getchar() function; putchar() function
    - changing, 265–266
    - comparisons, 265–266
  - sorting, 443–449
  - storage requirements, 248
  - terminating, 248
  - terminator character, 248–250
  - testing, functions for, 266
- strlen() (string length) function, 251, 256–257
- strncasecmp() (string comparison) function, 251
- strncat() (string concatenation) function, 251, 254–256
- strncpy() (string copy) function, 251
- strchr() (string character location) function, 251
- strstr() (string locate) function, 251
- struct keyword, 278–280
- struct variables. *See* structures; union variables
- structure variables, creating, 332–333
- structures. *See also* databases; linked lists; union variables
  - in arrays, 283–289
  - converting from arrays, 284–285
  - copying elements, 287–289
  - declaring, 281–283
  - input to functions, 308–310
  - nesting, 289–291
  - overview, 277–281
  - predefined data, 281–283, 285–287
  - reading and writing, sample, 566–574
  - reading from disk, 571
  - struct keyword, 278–280
  - typedef keyword, 279
  - Wizard of Oz* example, 272–277, 284–285
  - writing to disk, 569–570

subroutines. *See* functions; macros  
 subtraction  
   - (minus sign), subtraction, 110  
   -- (minus signs), decrement operator, 115, 121  
     decrementing by one, 115–119  
 Sun Microsystems, 10  
 switch-case statement  
   with no break, debugging, 513  
   overview, 205–209  
   while(!done) technique, 211–212  
 symbols. *See* operators  
 synchronizing files, 519  
 syntax errors, 43  
 system() (system) function, 541–542

---

## T

---

\t, tab character, 76  
 tab character, 76  
 tan() (tangent of an angle) function, 173–177  
 terminating programs. *See* ending programs  
 terminating strings, 248  
 terminator character, 248–250  
 text. *See also* strings  
   in float variables, 67  
   in int variables, 67  
   justifying, 163–164  
 text editors  
   case sensitivity, 25  
   Easy Editor, 674  
   MS-DOS Editor, 674  
   overview, 24–25  
   vim, 63, 674  
 text files *versus* binary, 554–559  
 text variables, 14  
 3-dimensional arrays, 239–240, 259–261  
 tilde (~), one's complement, 400–402  
 time. *See also* date and time  
   calculating a pause, 494–496  
   getting  
     current day, 491–492  
     as epoch value, 488–490  
     as formatted string, 490–491, 493–494  
   setting, 494  
 time() (time) function, 489–490  
 timestamping files, 519  
 tm structure, 492–493  
 toascii() (convert to ASCII) function, 266  
 tolower() (convert to lowercase) function, 266, 269–270  
 tool archives, 515  
 tools. *See* compilers; editors; linkers  
 touch program, 519  
 toupper() (convert to uppercase) function, 266  
 trigonometry, 171–177  
 trimming random numbers, 186–187  
 troubleshooting. *See* debugging  
 2-dimensional arrays, 236–239  
 typecasting variables, 146–148  
 typedef keyword  
   creating structure variables, 332–333  
   definition, 331  
   linked lists, 333–334  
   renaming variables, 146–148, 331–332  
   and structures, 279  
 types, variable data  
   changing temporarily  
     creating structure variables, 332–333  
     linked lists, 333–334  
     overview, 146–148  
     renaming variables, 331–332  
   structures, 279  
   list of, 14, 136–137, 767  
   mixing, 342–346  
   redeclaring, 146–148  
   treating one as another, 146–148

## 804 C All-in-One Desk Reference For Dummies

### U

`%u`, conversion character  
 definition, 771  
 sample program, 145–146

unary operators

- `&` (ampersand), address operator
  - initializing pointers, 355, 358, 365
  - in `scanf()` function, 358, 365
- `*` (asterisk), pointer operator
  - initializing pointers, 364
  - modifying variables, 376–380
  - previewing RAM contents, 374–376
  - requirements for, 361
  - syntax, 360
- `-` (minus sign), negative operator, 114–115
- `+` (plus sign), positive operator, 114–115
- `~` (tilde), one's complement, 400–402

overview, 114–115

- `sizeof` (return size of object), 353

underscore (`_`), in functions, 295

UNICODE, 65

union variables, 342–346. *See also* structures

Unix, history of C language, 10

Unix epoch, 487–488

`unlink()` (unlink file) function, 610

unsigned variables, 143–145

UTC (Coordinated Universal Time), 487

### V

values. *See also* constants; variables  
 definition, 23–24  
 getting from functions, 14–15  
 input to functions, 304–308  
 output width, setting, 161–163  
 retaining after use, 337–341  
 variable, assigning, 53–54, 58–59  
 variable, changing, 55–56

variables. *See also* constants; values  
 address, getting, 355–358  
 addresses, description, 355–358  
 addresses, getting, 358  
 addresses, pointers to, 360–364  
 assigning sequential values, 336–337  
`auto` keyword, 335  
`char`  
   declaring, 64  
   definition, 14, 137  
   as integer value, 65–67  
   single character, 62–63  
   size limitation, 67  
`const` keyword, 335  
 constants, creating, 335  
 data types  
   changing temporarily, 146–148.  
     *See also* `typedef` keyword  
   list of, 14, 136–137, 767  
   mixing, 342–346  
   redeclaring, 146–148  
   treating one as another, 146–148  
 declaring, 14, 52–53  
 definition, 14, 23–24  
`double`  
   definition, 14  
   very large numbers, 140–141  
 empty, 14  
`entry` keyword, 335  
`enum` keyword, 336–337  
 enumerating, 336–337  
`float`  
   definition, 14, 137  
   storing and displaying characters, 67  
   using, 56  
   very large/small values, 139–140  
 fractional numbers, 14  
 in functions  
   global, 300–303  
   local, 298–300

global, 300–303  
   programs with multiple modules, 501–503  
 initializing, 53–54  
 instead of constants, 103  
 int  
   definition, 14, 136  
   long, 67, 137–139  
   short, 67, 137–139  
   storing and displaying characters, 67  
   using, 54–55  
 integers, 14  
 list of, 148  
 local, 298–300  
 mathematical operations on  
   themselves, 120–121  
 multiple, 56–57  
 names  
   all caps, 332  
   Hungarian notation, 329–331  
   prefixes, 330  
   renaming, 331–332. *See also* typedef keyword  
   uniqueness, 306  
 naming conventions, 329–331  
 negative numbers, 141–146  
 order of precedence, 144  
 overflow, 138–139  
 placing in registers, 337  
 precision, 58, 140–141  
 register keyword, 337  
 retaining values after use, 337–341  
 sample program, 137  
 sharing, 501–503  
 signed, 142–143, 145  
 size, getting, 352–353  
 static keyword, 337–340  
 structure, creating, 332–333  
 text, 14  
 typecasting, 146–148

types  
   changing temporarily, 146–148.  
   *See also* typedef keyword  
   list of, 14, 136–137, 767  
   mixing, 342–346  
   redeclaring, 146–148  
   treating one as another, 146–148  
 union, 342–346. *See also* structures  
 unsigned, 143–145  
 values  
   assigning, 53–54, 58–59  
   changing, 55–56, 376–380  
 void, 14  
 volatile keyword, 341  
 vertical bar (|)  
   binary OR, 394–395, 397–398  
   piping output, 528–530  
 viewing. *See* displaying  
 vim editor, 63, 674  
 vocabulary, programmer, 511  
 void variable, 14  
 volatile keyword, 341

---

## W

Web resources. *See* online resources  
 while loops  
   (!done) technique, 197–199, 211–212  
   endless, 201–202  
   nested, 203–204  
   overview, 193–197  
 whole numbers (integers). *See also*  
   floating-point numbers  
   from ASCII values, 74–75  
   in char variables, 65–67  
   definition, 48–49  
 Windows compiler, 670–671  
 Wirth, Niklaus, 10  
 Wizard of Oz example, 272–277, 284–285  
 words, definition, 385

## 806 C All-in-One Desk Reference For Dummies

---

writing. *See also* input/output arrays to disk, 564  
databases to disk, 582–584  
files, overview, 547–550  
files, random access, 581–582  
formatted disk data, 562  
linked lists to disk, 644–645  
strings, 244–245. *See also* printf() function; puts() function  
structures, 566–574  
structures to disk, 569–570

---

### X

---

%X, conversion character  
definition, 771  
displaying hexadecimal numbers, 163

%x, conversion character  
definition, 771  
displaying hexadecimal numbers,  
153–154, 163  
\x, hexadecimal escape, 155

---

### Y

---

Yes-or-No puzzle, 123–127, 131

---

### Z

---

Zulu time, 487



