---

1. Declarations and Initializations

1.1 How do you decide which integer type to use?

1.4 What should the 64-bit type on new, 64-bit machines be?

1.7 What's the best way to declare and define global variables?

1.11 What does extern mean in a function declaration?

1.12 What's the auto keyword good for?

1.14 I can't seem to define a linked list node which contains a pointer to itself.

1.21 How do I declare an array of N pointers to functions returning pointers to functions returning pointers to characters?

1.22 How can I declare a function that returns a pointer to a function of its own type?

1.25 My compiler is complaining about an invalid redeclaration of a function, but I only define it once and call it once.

1.30 What can I safely assume about the initial values of variables which are not explicitly initialized?

1.31 Why can't I initialize a local array with a string?

1.32 What is the difference between char a[] = "string"; and char *p = "string"; ?

1.34 How do I initialize a pointer to a function?

---

2. Structures, Unions, and Enumerations

2.1 What's the difference between struct x1 { ... }; and typedef struct { ... } x2; ?

2.2 Why doesn't "struct x { ... }; x thestruct;" work?

2.3 Can a structure contain a pointer to itself?

2.4 What's the best way of implementing opaque (abstract) data types in C?

2.6 I came across some code that declared a structure with the last member an array of one element, and then did some tricky allocation to make it act like the array had several elements. Is this legal or portable?

2.7 I heard that structures could be assigned to variables and passed to and from functions, but K&R1 says not.

2.8 Why can't you compare structures?

2.9 How are structure passing and returning implemented?

2.10 Can I pass constant values to functions which accept structure arguments?

2.11 How can I read/write structures from/to data files?

2.12 How can I turn off structure padding?

2.13 Why does sizeof report a larger size than I expect for a structure type?

2.14 How can I determine the byte offset of a field within a structure?

2.15 How can I access structure fields by name at run time?

2.18 I have a program which works correctly, but dumps core after it finishes. Why?

2.20 Can I initialize unions?

2.22 What is the difference between an enumeration and a set of preprocessor #defines?

2.24 Is there an easy way to print enumeration values symbolically?

---

3. Expressions

3.1 Why doesn't the code "a[i] = i++;" work?

3.2 Under my compiler, the code "int i = 7; printf("%d\n", i++ * i++);" prints 49. Regardless of the order of evaluation, shouldn't it print 56?

---

4. Pointers

---

5. Null Pointers

---

6. Arrays and Pointers

6.3 So what is meant by the ``equivalence of pointers and arrays'' in C?

6.4 Why are array and pointer declarations interchangeable as function formal parameters?

6.7 How can an array be an lvalue, if you can't assign to it?

6.8 What is the real difference between arrays and pointers?

6.9 Someone explained to me that arrays were really just constant pointers.

6.11 I came across some ``joke'' code containing the ``expression'' 5["abcdef"] . How can this be legal C?

6.12 What's the difference between array and &array?

6.13 How do I declare a pointer to an array?

6.14 How can I set an array's size at compile time?

6.15 How can I declare local arrays of a size matching a passed-in array?

6.16 How can I dynamically allocate a multidimensional array?

6.17 Can I simulate a non-0-based array with a pointer?

6.18 My compiler complained when I passed a two-dimensional array to a function expecting a pointer to a pointer.

6.19 How do I write functions which accept two-dimensional arrays when the ``width'' is not known at compile time?

6.20 How can I use statically- and dynamically-allocated multidimensional arrays interchangeably when passing them to functions?

6.21 Why doesn't sizeof properly report the size of an array which is a parameter to a function?

---

7. Memory Allocation

7.1 Why doesn't the code ``char *answer; gets(answer);'' work?

7.2 I can't get strcat to work. I tried ``char *s3 = strcat(s1, s2);''
but I got strange results.

7.3 But the man page for strcat says that it takes two char *'s as
arguments. How am I supposed to know to allocate things?

7.5 I have a function that is supposed to return a string, but when it
returns to its caller, the returned string is garbage.

7.6 Why am I getting ``warning: assignment of pointer from integer
lacks a cast'' for calls to malloc?

:wq

7.7 Why does some code carefully cast the values returned by malloc to
the pointer type being allocated?

7.8 Why does so much code leave out the multiplication by sizeof(char)
when allocating strings?

7.14 I've heard that some operating systems don't actually allocate
malloc'ed memory until the program tries to use it. Is this legal?

7.16 I'm allocating a large array for some numeric work, but malloc is
acting strangely.

7.17 I've got 8 meg of memory in my PC. Why can I only seem to malloc
640K or so?

7.19 My program is crashing, apparently somewhere down inside malloc.

7.20 You can't use dynamically-allocated memory after you free it, can
you?

7.21 Why isn't a pointer null after calling free?

7.22 When I call malloc to allocate memory for a local pointer, do I
have to explicitly free it?

7.23 When I free a dynamically-allocated structure containing
pointers, do I have to free each subsidiary pointer first?

7.24 Must I free allocated memory before the program exits?

7.25 Why doesn't my program's memory usage go down when I free memory?

---

11. ANSI/ISO Standard C

11.1 What is the ``ANSI C Standard?''

11.2 How can I get a copy of the Standard?

11.3 My ANSI compiler is complaining about prototype mismatches for parameters declared float.

11.4 Can you mix old-style and new-style function syntax?

11.5 Why does the declaration "extern f(struct x *p);" give me a warning message?

11.8 Why can't I use const values in initializers and array dimensions?

11.9 What's the difference between const char *p and char * const p?

11.10 Why can't I pass a char ** to a function which expects a const char **?

11.12 Can I declare main as void, to shut off these annoying ``main returns no value'' messages?

11.13 But what about main's third argument, envp?

11.14 I believe that declaring void main() can't fail, since I'm calling exit instead of returning.

11.15 The book I've been using always uses void main().

11.16 Is exit(status) truly equivalent to returning the same status from main?

11.17 How do I get the ANSI ``stringizing'' preprocessing operator `#' to stringize the macro's value instead of its name?

11.18 What does the message ``warning: macro replacement within a string literal'' mean?

11.19 I'm getting strange syntax errors inside lines I've #ifdeffed out.

11.20 What are #pragmas ?

-- pre 11.21 What does ``#pragma once'' mean?

11.22 Is char a[3] = "abc"; legal?

---

12. Stdio

12.6 How can I print a '%' character with printf?

12.9 How can printf use %f for type double, if scanf requires %lf?

12.10 How can I implement a variable field width with printf?

12.11 How can I print numbers with commas separating the thousands?

12.12 Why doesn't the call scanf("%d", i) work?

-- 12.13 Why doesn't the code "double d; scanf("%f", &d);" work?

12.15 How can I specify a variable width in a scanf format string?

12.17 When I read numbers from the keyboard with scanf "%d\n", it seems to hang until I type one extra line of input.

12.18 I'm reading a number with scanf %d and then a string with gets(), but the compiler seems to be skipping the call to gets()!

12.19 I'm re-prompting the user if scanf fails, but sometimes it seems to go into an infinite loop.

12.20 Why does everyone say not to use scanf? What should I use instead?

12.21 How can I tell how much destination buffer space I'll need for an arbitrary sprintf call? How can I avoid overflowing the destination buffer with sprintf?

12.23 Why does everyone say not to use gets()?

12.24 Why does errno contain ENOTTY after a call to printf?

12.25 What's the difference between fgetpos/fsetpos and ftell/fseek?

12.26 Will fflush(stdin) flush unread characters from the standard input stream?

12.30 I'm trying to update a file in place, by using fopen mode "r+", but it's not working.

12.33 How can I redirect stdin or stdout from within a program?

12.34 Once I've used freopen, how can I get the original stream back?

12.38 How can I read a binary data file properly?

---

13. Library Functions

13.1 How can I convert numbers to strings?

13.2 Why does strncpy not always write a '\0'?

13.5 Why do some versions of toupper act strangely if given an upper-case letter?

13.6 How can I split up a string into whitespace-separated fields?

13.7 I need some code to do regular expression and wildcard matching.

13.8 I'm trying to sort an array of strings with qsort, using strcmp as the comparison function, but it's not working.

13.9 Now I'm trying to sort an array of structures, but the compiler is complaining that the function is of the wrong type for qsort.

13.10 How can I sort a linked list?

13.11 How can I sort more data than will fit in memory?

13.12 How can I get the time of day in a C program?

13.13 How can I convert a struct tm or a string into a time_t?

13.14 How can I perform calendar manipulations?

13.15 I need a random number generator.

13.16 How can I get random integers in a certain range?

13.17 Each time I run my program, I get the same sequence of numbers back from rand().

13.18 I need a random true/false value, so I'm just taking rand() % 2, but it's alternating 0, 1, 0, 1, 0...

13.20 How can I generate random numbers with a normal or Gaussian distribution?

13.24 I'm trying to port this old program. Why do I get ``undefined external'' errors for some library functions?

13.25 I get errors due to library functions being undefined even though I #include the right header files.

13.26 I'm still getting errors due to library functions being

undefined, even though I'm requesting the right libraries.

13.28 What does it mean when the linker says that _end is undefined?

---

14. Floating Point

14.1 When I set a float variable to 3.1, why is printf printing it as 3.0999999?

14.2 Why is sqrt(144.) giving me crazy numbers?

14.3 I keep getting ``undefined: sin'' compilation errors.

14.4 My floating-point calculations are acting strangely and giving me different answers on different machines.

14.5 What's a good way to check for ``close enough'' floating-point equality?

14.6 How do I round numbers?

14.7 Where is C's exponentiation operator?

14.8 The pre-#defined constant M_PI seems to be missing from <math.h>.

14.9 How do I test for IEEE NaN and other special values?

14.11 What's a good way to implement complex numbers in C?

14.12 I'm looking for some mathematical library code.

14.13 I'm having trouble with a Turbo C program which crashes and says something like ``floating point formats not linked.''

---

15. Variable-Length Argument Lists

15.1 I heard that you have to #include <stdio.h> before calling printf. Why?

15.2 How can %f be used for both float and double arguments in printf?

15.3 Why don't function prototypes guard against mismatches in printf's arguments?

15.4 How can I write a function that takes a variable number of arguments?

15.5 How can I write a function that takes a format string and a variable number of arguments, like printf, and passes them to printf to do most of the work?

15.6 How can I write a function analogous to scanf, that calls scanf to do most of the work?

15.7 I have a pre-ANSI compiler, without <stdarg.h>. What can I do?

15.8 How can I discover how many arguments a function was actually called with?

15.9 My compiler isn't letting me declare a function that accepts only variable arguments.

15.10 Why isn't "va_arg(argp, float)" working?

15.11 I can't get va_arg to pull in an argument of type pointer-to-function.

15.12 How can I write a function which takes a variable number of arguments and passes them to some other function ?

15.13 How can I call a function with an argument list built up at run time?

---

16. Strange Problems

16.3 This program crashes before it even runs!

16.4 I have a program that seems to run correctly, but then crashes as it's exiting.

16.5 This program runs perfectly on one machine, but I get weird results on another.

16.6 Why does the code "char *p = "hello, world!"; p[0] = 'H';" crash?

16.8 What does ``Segmentation violation'' mean?

---

17. Style

17.1 What's the best style for code layout in C?

17.3 Is the code "if(!strcmp(s1, s2))" good style?

17.4 Why do some people write if(0 == x) instead of if(x == 0)?

17.5 I came across some code that puts a (void) cast before each call to printf. Why?

17.8 What is Hungarian Notation''? Is it worthwhile?

17.9 Where can I get the ``Indian Hill Style Guide'' and other coding standards?

17.10 Some people say that goto's are evil and that I should never use them. Isn't that a bit extreme?

---

18. Tools and Resources

18.1 I'm looking for C development tools (cross-reference generators, code beautifiers, etc.).

18.2 How can I track down these pesky malloc problems?

18.3 What's a free or cheap C compiler I can use?

18.4 I just typed in this program, and it's acting strangely. Can you see anything wrong with it?

18.5 How can I shut off the ``warning: possible pointer alignment problem'' message which lint gives me for each call to malloc?

18.7 Where can I get an ANSI-compatible lint?

18.8 Don't ANSI function prototypes render lint obsolete?

18.9 Are there any C tutorials or other resources on the net?

18.10 What's a good book for learning C?

18.13 Where can I find the sources of the standard C libraries?

18.14 I need code to parse and evaluate expressions.

18.15 Where can I get a BNF or YACC grammar for C?

18.15a Does anyone have a C compiler test suite I can use?

18.16 Where and how can I get copies of all these freely distributable programs?

---

19. System Dependencies

19.1 How can I read a single character from the keyboard without waiting for the RETURN key?

19.2 How can I find out how many characters are available for reading, or do a non-blocking read?

19.3 How can I display a percentage-done indication that updates itself in place, or show one of those ``twirling baton'' progress indicators?

19.4 How can I clear the screen, or print things in inverse video, or move the cursor?

19.5 How do I read the arrow keys? What about function keys?

19.6 How do I read the mouse?

19.7 How can I do serial (``comm'') port I/O?

19.8 How can I direct output to the printer?

19.9 How do I send escape sequences to control a terminal or other device?

19.10 How can I do graphics?

19.11 How can I check whether a file exists?

19.12 How can I find out the size of a file, prior to reading it in?

19.13 How can a file be shortened in-place without completely clearing or rewriting it?

19.14 How can I insert or delete a line in the middle of a file?

19.15 How can I recover the file name given an open file descriptor?

19.16 How can I delete a file?

19.17 What's wrong with the call "fopen("c:\newdir\file.dat", "r")"?

19.18 How can I increase the allowable number of simultaneously open files?

19.20 How can I read a directory in a C program?

20.8 How can I implement sets or arrays of bits?

20.9 How can I determine whether a machine's byte order is big-endian or little-endian?

20.10 How can I convert integers to binary or hexadecimal?

20.11 Can I use base-2 constants (something like 0b101010)? Is there a printf format for binary?

20.12 What is the most efficient way to count the number of bits which are set in a value?

20.13 How can I make my code more efficient?

20.14 Are pointers really faster than arrays? How much do function calls slow things down?

20.17 Is there a way to switch on strings?

20.18 Is there a way to have non-constant case labels (i.e. ranges or arbitrary expressions)?

20.19 Are the outer parentheses in return statements really optional?

20.20 Why don't C comments nest? Are they legal inside quoted strings?

20.24 Why doesn't C have nested functions?

-

20.25 How can I call FORTRAN (C++, BASIC, Pascal, Ada, LISP) functions from C?

20.26 Does anyone know of a program for converting Pascal or FORTRAN to C?

20.27 Can I use a C++ compiler to compile C code?

20.28 I need to compare two strings for close, but not necessarily exact, equality.

20.29 What is hashing?

20.31 How can I find the day of the week given the date?

20.32 Will 2000 be a leap year?

20.34 How do you write a program which produces its own source code as

its output?

20.35 What is ``Duff's Device''?

20.36 When will the next Obfuscated C Code Contest be held? How can I get a copy of previous winning entries?

20.37 What was the entry keyword mentioned in K&R1?

20.38 Where does the name ``C'' come from, anyway?

20.39 How do you pronounce ``char''?

20.40 Where can I get extra copies of this list?