9 Essential C Interview Questions *

What will be the output when the following code is executed? Explain.

```
#include <stdio.h>
#define SQUARE(a) (a)*(a)

int main() {
    printf("%d\n", SQUARE(4));
    int x = 3;
    printf("%d\n", SQUARE(++x));
}
```

View the answer → Hide answer

The answer is infact undefined, and depends on the compiler being used. Some compilers will result in `16` and `20`, while others will produce `16` and `25`.

One might expect the second use of the `SQUARE` macro to yield `16`, just like the first use of the `SQUARE` macro. However, macros are processed by the preprocessor, a step that takes place before actual compilation begins. Expanding the second macro will show what actually gets compiled:

```
(++x)*(++x)
```

The evaluation of the pre-increment operation `++x` is where the undefined behavior in C comes in. With some compilers, the macro will reduce to `(4)*(5)`, while in other cases, it will be evaluated as `(5)*(5)`.

This article discusses this behavior further.

Why is it usually a bad idea to use `gets()`? Suggest a workaround.

View the answer →Hide answer

The function `gets()` reads characters from the `stdin` and stores them at the provided input buffer. However, `gets()` will keep reading until it encounters a newline character. Unless the buffer is large enough, or the length of the line being read is known ahead of time, `gets()` can potentially overflow the input buffer and start overwriting memory it is not supposed to, wreaking havoc or opening security vulnerabilities.

One way to work around this issue is to use `fgets()`. It allows you to put a limit on the maximum number of characters to read:

```
fgets(b, 124, stdin);
```

What is the difference between structs and unions?

View the answer →Hide answer

A struct is a complex data type that allows multiple variables to be stored in a group at a named block of memory. Each member variable of a struct can store different data, and they all can be used at once.

```
struct a {
    int x;
    char y;
} a;
```

For example, you may store an integer in `x`, and and a character in `y` above, independent of each other.

A union, on the other hand, stores the contents of any member variable at the exact same memory location. This allows the storage of different types of data at the same memory location. The result is that assigning a value to one member will change the value of all the other members. Unlike struct, only one member of the union type is likely to be useful at any given time.

```
union b {
    int x;
    char y;
} b;
```

For example, storing a character in `y` may automatically change the integer you read from `x` to something meaningless or unpredictable.

Find top C developers today. Toptal can match you with the best engineers to finish your project.

Hire Toptal's C developers

What is a void pointer? Can you dereference a void pointer without knowing its type?

View the answer →Hide answer

A void pointer is a pointer that can be used to point to any data of any arbitrary type. A void pointer can be dereferenced only after explicit casting. For example:

```
int a = 5;
void *b = &a;
printf("%d\n", *((int*)b));
```

What is being declared in the following statement?

```
char (*x) (char*);
```

View the answer →Hide answer

The statement above declares `x` as a pointer to a function that takes a single character-pointer argument, and returns a character.

What is the difference between `#include "..."` and `#include <...>`?

View the answer →Hide answer

The difference lies in where the preprocessor looks for the file to be included. For the `include` directive with a double quoted filename, the preprocessor limits its search for the file to the same directory where the current source file resides, and then to the standard directories pre-designated by the compiler. On the other hand, when the directive uses angle brackets, the preprocessor searches for the file in directories pre-designated by the compiler - usually directories where standard library header files reside.

What are dangling pointers? How are dangling pointers different from memory leaks?

View the answer →Hide answer

Dangling pointers are those that point to memory locations which have already been freed. For example:

```
int *a = malloc(sizeof(int));
free(a);
// a is now a dangling pointer
```

Memory leaks are quite the opposite of dangling pointers. Memory leaks happen when memory locations are not freed, but there is no way to refer to them (i.e., no pointers are pointing to them).

```
int *a = malloc(sizeof(int));
a = 0;
// now a no longer points to the memory that we just allocated, causing a memory leak
```

Unlike higher-level languages with garbage collectors, it is critical to always keep track of allocated memory when programming in C.

This code snippet converts a floating point number to an integer using casting:

```
float f = 1.0;

int i1 = (int) f;
int i2 = * (int *) &f;

printf("%d\n, i1);
printf("%d\n, i2);
```

The following output is produced:

```
1
1065353216
```

Can you explain why results differ?

View the answer →Hide answer

The first casting operation properly converts from a floating point number to an integer, as specified by the C standard. The second conversion, however, is first casting a float pointer to an integer pointer which is then dereferenced to get the final result. This way the compiler is effectively treating raw bits from a float (typically stored in IEEE floating point format) as if they were bits of an integer. Besides getting a wrong result you are potentially doing a "bad read" operation, in cases where `sizeof(int)` is greater than `sizeof(float)` (e.g. on some 64-bit architectures).

Although this particular code is unlikely, it demonstrates one of the risks involved in typecasting when only a pointer to the variable to be cast is available. In practice, the pointer must be dereferenced *before* it is cast.

What is the output of the following program if run on a 32-bit operating system?

```
#include <stdio.h>
int  main()
{
  int a=-2;
  printf("%x",a>>3);
}
```

View the answer →Hide answer

In a 32 bit operating system, integers are stored as 4 bytes.

Since a is negative, it will be stored in 2's complement. When an integer is negative and we want to right shift by "n" bits, we need to prepend ones (not zeros!) to the left hand side. The answer would therefore be `0xFFFF` (`%x` prints out value in hex).

* There is more to interviewing than tricky technical questions, so these are intended merely as a guide. Not every "A" candidate worth hiring will be able to answer them all, nor does answering them all guarantee an "A" candidate. At the end of the day, hiring remains an art, a science — and a lot of work.
Submit an interview question
Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Name

Email

Enter your question here

Enter your answer here

All fields are required

I agree with the Terms and Conditions of Toptal, LLC's Privacy Policy

Submit a Question

Thanks for submitting your question.
Our editorial staff will review it shortly. Please note that submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.
Looking for C experts? Check out Toptal's C developers.

View full profile »
Michael Truog
United States
Michael is a top architect, engineer, developer, and entrepreneur with a proven ability to develop efficient, scalable, and fault-tolerant server solutions for complex problems. He has extensive experience and skills with all levels of software and architecture.
CJavaRubyErlangPython+3 more
Hire Michael

View full profile »
Dmitrii Polutov
Australia
Dmitrii is a Software Engineer with a strong background in the development, design, and maintenance of new and existing software. He has extensive experience programming across multiple platforms, writing C and C++ code for over two decades.
CC++UI KitCocoaXcode+2 more
Hire Dmitrii

View full profile »
Richard Rozsa
Netherlands
Richard Rozsa offers a vision of data as a self formatting entity. For more than 30 years, he's delivered top quality technical architecture, programming, testing and solutions for complex problems--on-time and within budget. He's extremely flexible and able to integrate as a standalone freelancer or within teams.
C.NETASP.NETASP.NET MVC+9 more
Hire Richard

Toptal connects the top 3% of freelance talent all over the world.

Join the Toptal community.

Hire a developer
or
Apply as a developer

Highest In-Demand Talent

- iOS Developers
- Front-End Developers
- UX Designers
- UI Designers
- Financial Modeling Consultants
- Interim CFOs
- Digital Project Managers

About

- Top 3%
- Clients
- Freelance Developers
- Freelance Designers
- Freelance Finance Experts
- Freelance Project Managers
- Freelance Product Managers
- About Us

Contact

- Contact Us
- Press Center
- Careers
- FAQ

Social

Hire the top 3% of freelance talent™