`offsetof` computes the number of bytes between the beginning of the structure and the specified member.

Consider the following structure:

```
struct s {
  char a;
  int b[2];
  float c;
};
```

The value of `offsetof(struct s, a)` must be 0: C guarantees that the first member of a structure has the same address as the structure itself. We can't say for sure what the offsets of `b` and `c` are. One possibility is that `offsetof(struct s, b)` is 1 (since `a` is one byte long), and `offsetof(struct s, c)` is 9 (assuming 32-bit integers). However, some compilers leave "holes"—unused bytes—in structures (see the Q&A section at the end of Chapter 16), which can affect the value produced by `offsetof`. If a compiler should leave a three-byte hole after `a`, for example, then the offsets of `b` and `c` would be 4 and 12, respectively. But that's the beauty of `offsetof`: it produces the correct offsets for any compiler, enabling us to write portable programs.

There are various uses for `offsetof`. For example, suppose that we want to save the first two members of an `s` structure in a file, ignoring the `c` member. Instead of having the `fwrite` function write `sizeof(struct s)` bytes, which would save the entire structure, we'll tell it to write only `offsetof(struct s, c)` bytes.

`fwrite` function ➤22.6

A final remark: Some of the types and macros defined in `<stddef.h>` appear in other headers as well. (The `NULL` macro, for example, is also defined in `<locale.h>`, `<stdio.h>`, `<stdlib.h>`, `<string.h>`, and `<time.h>`, as well as in the C99 header `<wchar.h>`.) As a result, few programs need to include `<stddef.h>`.

21.5 The `<stdbool.h>` Header (C99): Boolean Type and Values

The `<stdbool.h>` header defines four macros:

- `bool` (defined to be `_Bool`)
- `true` (defined to be `1`)
- `false` (defined to be `0`)
- `__bool_true_false_are_defined` (defined to be `1`)

We've seen many examples of how `bool`, `true`, and `false` are used. Potential uses of the `__bool_true_false_are_defined` macro are more limited. A program could use a preprocessing directive (such as `#if` or `#ifdef`) to test this macro before attempting to define its own version of `bool`, `true`, or `false`.