- ***Temporarily disabling code that contains comments.*** We can't use a `/*...*/` comment to "comment out" code that already contains `/*...*/` comments. Instead, we can use an `#if` directive:

```
#if 0
Lines containing comments
#endif
```

Q&A

Disabling code in this way is often called "conditioning out."

Section 15.2 discusses another common use of conditional compilation: protecting header files against multiple inclusion.

14.5 Miscellaneous Directives

To end the chapter, we'll take a brief look at the `#error`, `#line`, and `#pragma` directives. These directives are more specialized than the ones we've already examined, and they're used much less frequently.

The `#error` Directive

The `#error` directive has the form

`#error` directive

```
#error message
```

where *message* is any sequence of tokens. If the preprocessor encounters an `#error` directive, it prints an error message which must include *message*. The exact form of the error message can vary from one compiler to another; it might be something like

```
Error directive: message
```

or perhaps just

```
#error message
```

Encountering an `#error` directive indicates a serious flaw in the program; some compilers immediately terminate compilation without attempting to find other errors.

`#error` directives are frequently used in conjunction with conditional compilation to check for situations that shouldn't arise during a normal compilation. For example, suppose that we want to ensure that a program can't be compiled on a machine whose `int` type isn't capable of storing numbers up to 100,000. The largest possible `int` value is represented by the `INT_MAX` macro, so all we need do is invoke an `#error` directive if `INT_MAX` isn't at least 100,000:

`INT_MAX` macro ➤*23.2*