known as ***macro definition***, we can name this constant:

```
#define INCHES_PER_POUND 166
```

`#define` is a preprocessing directive, just as `#include` is, so there's no semicolon at the end of the line.

When a program is compiled, the preprocessor replaces each macro by the value that it represents. For example, the statement

```
weight = (volume + INCHES_PER_POUND - 1) / INCHES_PER_POUND;
```

will become

```
weight = (volume + 166 - 1) / 166;
```

giving the same effect as if we'd written the latter statement in the first place.

The value of a macro can be an expression:

```
#define RECIPROCAL_OF_PI (1.0f / 3.14159f)
```

parentheses in macros ➤ 14.3

If it contains operators; the expression should be enclosed in parentheses.

Notice that we've used only upper-case letters in macro names. This is a convention that most C programmers follow, not a requirement of the language. (Still, C programmers have been doing this for decades; you wouldn't want to be the first to deviate.)

PROGRAM

Converting from Fahrenheit to Celsius

The following program prompts the user to enter a Fahrenheit temperature; it then prints the equivalent Celsius temperature. The output of the program will have the following appearance (as usual, input entered by the user is underlined):

```
Enter Fahrenheit temperature: 212
Celsius equivalent: 100.0
```

The program will allow temperatures that aren't integers; that's why the Celsius temperature is displayed as `100.0` instead of `100`. Let's look first at the entire program, then see how it's put together.

celsius.c

```
/* Converts a Fahrenheit temperature to Celsius */

#include <stdio.h>

#define FREEZING_PT 32.0f
#define SCALE_FACTOR (5.0f / 9.0f)

int main(void)
{
  float fahrenheit, celsius;

  printf("Enter Fahrenheit temperature: ");
```