```
int main(void)
{
  ...
}
```

Older C programs often omit `main`'s return type, taking advantage of the fact that it traditionally defaults to `int`:

```
main()
{
  ...
}
```

C99 Omitting the return type of a function isn't legal in C99, so it's best to avoid this practice. Omitting the word `void` in `main`'s parameter list remains legal, but—as a matter of style—it's best to be explicit about the fact that `main` has no parameters. (We'll see later that `main` sometimes *does* have two parameters, usually named `argc` and `argv`.)

`argc` and `argv` ➤ *13.7*

Q&A The value returned by `main` is a status code that—in some operating systems—can be tested when the program terminates. `main` should return 0 if the program terminates normally; to indicate abnormal termination, `main` should return a value other than 0. (Actually, there's no rule to prevent us from using the return value for other purposes.) It's good practice to make sure that every C program returns a status code, even if there are no plans to use it, since someone running the program later may decide to test it.

The `exit` Function

Executing a `return` statement in `main` is one way to terminate a program. Another is calling the `exit` function, which belongs to `<stdlib.h>`. The argument passed to `exit` has the same meaning as `main`'s return value: both indicate the program's status at termination. To indicate normal termination, we'd pass 0:

`<stdlib.h>` header ➤ *26.2*

```
exit(0);                    /* normal termination */
```

Since 0 is a bit cryptic, C allows us to pass `EXIT_SUCCESS` instead (the effect is the same):

```
exit(EXIT_SUCCESS);    /* normal termination */
```

Passing `EXIT_FAILURE` indicates abnormal termination:

```
exit(EXIT_FAILURE);    /* abnormal termination */
```

`EXIT_SUCCESS` and `EXIT_FAILURE` are macros defined in `<stdlib.h>`. The values of `EXIT_SUCCESS` and `EXIT_FAILURE` are implementation-defined; typical values are 0 and 1, respectively.

As methods of terminating a program, `return` and `exit` are closely related. In fact, the statement

`return` *expression*`;`