Another way to argue with a function

This book shows you the modern, convenient way of declaring variables (or *arguments*) shuffled off to a function. To wit:

```
void jerk(int repeat, char c);
{
and so on. . . .
```

You can also use the original format:

```
void jerk(repeat, c)
int repeat;
char c;
{
and so on...
```

This declaration does the same thing, but it's a little more confusing because the variable name is introduced first and then the "what it is declaration" comes on the following line (or lines). Otherwise, the two are the same.

My advice is to stick with the format used in this book and try not to be alarmed if you see the other format used. Older C references may use the second format, and certain fogey C programmers may adhere to it. Beware!

Functions That Return Stuff

For some functions to properly funct, they must return a value. You pass along your birthday, and the function magically tells you how old you are (and then the computer giggles at you). This process is known as returning a value, and a heck of a lot of functions do that.

Something for your troubles

To return a value, a function must obey these two rules:

Warning! Rules approaching.

- The function has to be *defined* as a certain type (`int`, `char`, or `float`, for example — just like a variable). Use something other than `void`.
- The function has to return a value.

The function type tells you what type of value it returns. For example:

```
int birthday(int date);
```

The function `birthday()` is defined on this line. It's an integer function and returns an integer value. (It also requires an integer parameter, `date`, which it uses as input.)