Preprocessor substitution converts the declaration to this:

```c
float * pa, pb; // pa a pointer to float, pb just a float
```

The `typedef` approach doesn't have that problem.

Notice that `typedef` doesn't create a new type. It just creates a new name for an old type. If you make `word` an alias for `int`, `cout` treats a type `word` value as the `int` it really is.

**The do while Loop**

You've now seen the `for` loop and the `while` loop. The third C++ loop is the `do while`. It's different from the other two because it's an `exit-condition` loop. That means this devil-may-care loop first executes the body of the loop and only then evaluates the test expression to see whether it should continue looping. If the condition evaluates to `false`, the loop terminates; otherwise, a new cycle of execution and testing begins. Such a loop always executes at least once because its program flow must pass through the body of the loop before reaching the test. Here's the syntax:

```c
do
    body
while (test-expression);
```

The body portion can be a single statement or a brace-delimited statement block. Figure 5.4 summarizes the program flow for the `do while` loop.

**Figure 5.4. The do while loop.**