cin >> int toad;    // can't combine declaration with cin

Similarly, you can't assign a for loop to a variable:

int fx = for (int i = 0; i< 4; i++)
   cout >> i;    // not possible

Here the for loop is not an expression, so it has no value and you can't assign it.

Bending the Rules

C++ adds a feature to C loops that requires some artful adjustments to the for loop syntax. This was the original syntax:

for (expression; expression; expression)
   statement

In particular, the control section of a for structure consisted of three expressions, as defined earlier, separated by semicolons. C++ loops allow you to do things like the following, however:

for (int i = 0; i < 5; i++)

That is, you can declare a variable in the initialization area of a for loop. This can be convenient, but it doesn't fit the original syntax because a declaration is not an expression. This lawless behavior originally was accommodated by defining a new kind of expression, the declaration-statement expression, which was a declaration stripped of the semicolon, and which could appear only in a for statement. That adjustment has been dropped, however. Instead, the syntax for the for statement has been modified to the following:

for (for-init-statement condition; expression)
   statement

At first glance, this looks odd because there is just one semicolon instead of two. But that's okay because the for-init-statement is identified as a statement, and a statement has its own semicolon. As for the for-init-statement, it's identified as either an expression-statement or a declaration. This syntax rule replaces an expression followed by a