int x = 100; // new x
    cout << x << "\n"; // use new x
} // block ends
cout << x << "\n"; // use original x
return 0;

### The Comma Operator (or More Syntax Tricks)

The block, as you saw, enables you to sneak two or more statements into a place where C++ syntax allows just one statement. The comma operator does the same for expressions, enabling you to sneak two expressions into a place where C++ syntax allows only one expression. For example, suppose you have a loop in which one variable increases by 1 each cycle and a second variable decreases by 1 each cycle. Doing both in the update part of a for loop control section would be convenient, but the loop syntax allows just one expression there. The solution is to use the comma operator to combine the two expressions into one:

```
j++, i--; // two expressions count as one for syntax purposes
```

The comma is not always a comma operator. For example, the comma in the declaration

```
int i, j; // comma is a separator here, not an operator
```

serves to separate adjacent names in a list of variables.

Listing 5.9 uses the comma operator twice in a program that reverses the contents of a character array. Note that Listing 5.6 displays the contents of an array in reverse order, but Listing 5.9 actually moves characters around in the array. The program also uses a block to group several statements into one.

### Listing 5.9 forstr2.cpp

```
// forstr2.cpp -- reversing an array
#include <iostream>
#include <cstring>
```