array, then `++pt` changes `pt` so that it points to the second member.

The increment and decrement operators are nifty little operators, but don't get carried away and increment or decrement the same value more than once in the same statement. The problem is that the use-then-change and change-then-use rules can become ambiguous. That is, a statement such as

```c
x = 2 * x++ * (3 - ++x); // don't do it
```

can produce quite different results on different systems. C++ does not define correct behavior for this sort of statement.

**Combination Assignment Operators**

Listing 5.5 uses the following expression to update a loop counter:

```c
i = i + by
```

C++ has a combined addition and assignment operator that accomplishes the same result more concisely:

```c
i += by
```

The `+=` operator adds the values of its two operands and assigns the result to the operand on the left. This implies that the left operand must be something to which you can assign a value, such as a variable, an array element, a structure member, or data you identify by dereferencing a pointer:

```c
int k = 5;
k += 3; // ok, k set to 8
int *pa = new int[10]; // pa points to pa[0]
pa[4] = 12;
*(pa + 4) += 7; // ok, pa[4] set to 25
pa += 2; // ok, pa points to the former pa[2]
34 += 10; // quite wrong
```