program will still compile. Unfortunately, the latter expression is equivalent to `i = (+j)`, which merely copies the value of `j` into `i`.

The compound assignment operators have the same properties as the = operator. In particular, they're right associative, so the statement

```
i += j += k;
```

means

```
i += (j += k);
```

4.3 Increment and Decrement Operators

Two of the most common operations on a variable are "incrementing" (adding 1) and "decrementing" (subtracting 1). We can, of course, accomplish these tasks by writing

```
i = i + 1;
j = j - 1;
```

The compound assignment operators allow us to condense these statements a bit:

```
i += 1;
j -= 1;
```

But C allows increments and decrements to be shortened even further, using the `++` (*increment*) and `--` (*decrement*) operators.

Q&A

At first glance, the increment and decrement operators are simplicity itself: `++` adds 1 to its operand, whereas `--` subtracts 1. Unfortunately, this simplicity is misleading—the increment and decrement operators can be tricky to use. One complication is that `++` and `--` can be used as *prefix* operators (`++i` and `--i`, for example) or *postfix* operators (`i++` and `i--`). The correctness of a program may hinge on picking the proper version.

Another complication is that, like the assignment operators, `++` and `--` have side effects: they modify the values of their operands. Evaluating the expression `++i` (a "pre-increment") yields `i + 1` and—as a side effect—increments `i`:

```
i = 1;
printf("i is %d\n", ++i);    /* prints "i is 2" */
printf("i is %d\n", i);      /* prints "i is 2" */
```

Evaluating the expression `i++` (a "post-increment") produces the result `i`, but causes `i` to be incremented afterwards:

```
i = 1;
printf("i is %d\n", i++);    /* prints "i is 1" */
printf("i is %d\n", i);      /* prints "i is 2" */
```