

# 4 Expressions

*One does not learn computing by using a hand calculator, but one can forget arithmetic.*

One of C's distinguishing characteristics is its emphasis on expressions—formulas that show how to compute a value—rather than statements. The simplest expressions are variables and constants. A variable represents a value to be computed as the program runs; a constant represents a value that doesn't change. More complicated expressions apply operators to operands (which are themselves expressions). In the expression  $a + (b * c)$ , the  $+$  operator is applied to the operands  $a$  and  $(b * c)$ , both of which are expressions in their own right.

Operators are the basic tools for building expressions, and C has an unusually rich collection of them. To start off, C provides the rudimentary operators that are found in most programming languages:

- Arithmetic operators, including addition, subtraction, multiplication, and division.
- Relational operators to perform comparisons such as “ $i$  is *greater than* 0.”
- Logical operators to build conditions such as “ $i$  is greater than 0 *and*  $i$  is less than 10.”

But C doesn't stop here; it goes on to provide dozens of other operators. There are so many operators, in fact, that we'll need to introduce them gradually over the first twenty chapters of this book. Mastering so many operators can be a chore, but it's essential to becoming proficient at C.

In this chapter, we'll cover some of C's most fundamental operators: the arithmetic operators (Section 4.1), the assignment operators (Section 4.2), and the increment and decrement operators (Section 4.3). Section 4.1 also explains operator precedence and associativity, which are important for expressions that contain more than one operator. Section 4.4 describes how C expressions are evaluated. Finally, Section 4.5 introduces the expression statement, an unusual feature that allows any expression to serve as a statement.