6 Loops

A program without a loop and a structured variable isn't worth writing.

Chapter 5 covered C's selection statements, `if` and `switch`. This chapter introduces C's iteration statements, which allow us to set up loops.

A ***loop*** is a statement whose job is to repeatedly execute some other statement (the ***loop body***). In C, every loop has a ***controlling expression***. Each time the loop body is executed (an ***iteration*** of the loop), the controlling expression is evaluated; if the expression is true—has a value that's not zero—the loop continues to execute.

C provides three iteration statements: `while`, `do`, and `for`, which are covered in Sections 6.1, 6.2, and 6.3, respectively. The `while` statement is used for loops whose controlling expression is tested *before* the loop body is executed. The `do` statement is used if the expression is tested *after* the loop body is executed. The `for` statement is convenient for loops that increment or decrement a counting variable. Section 6.3 also introduces the comma operator, which is used primarily in `for` statements.

The last two sections of this chapter are devoted to C features that are used in conjunction with loops. Section 6.4 describes the `break`, `continue`, and `goto` statements. `break` jumps out of a loop and transfers control to the next statement after the loop, `continue` skips the rest of a loop iteration, and `goto` jumps to any statement within a function. Section 6.5 covers the null statement, which can be used to create loops with empty bodies.

6.1 The `while` Statement

Of all the ways to set up loops in C, the `while` statement is the simplest and most fundamental. The `while` statement has the form