11 Pointers

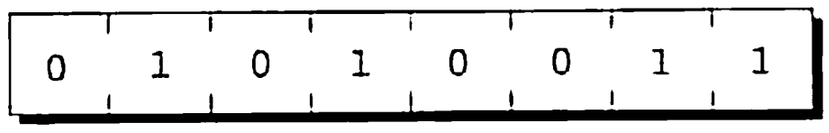
The 11th commandment was "Thou Shalt Compute"
or "Thou Shalt Not Compute"—I forget which.

Pointers are one of C's most important—and most often misunderstood—features. Because of their importance, we'll devote three chapters to pointers. In this chapter, we'll concentrate on the basics; Chapters 12 and 17 cover more advanced uses of pointers.

We'll start with a discussion of memory addresses and their relationship to pointer variables (Section 11.1). Section 11.2 then introduces the address and indirection operators. Section 11.3 covers pointer assignment. Section 11.4 explains how to pass pointers to functions, while Section 11.5 discusses returning pointers from functions.

11.1 Pointer Variables

The first step in understanding pointers is visualizing what they represent at the machine level. In most modern computers, main memory is divided into *bytes*, with each byte capable of storing eight bits of information:



Each byte has a unique ***address*** to distinguish it from the other bytes in memory. If there are n bytes in memory, we can think of addresses as numbers that range from 0 to $n - 1$ (see the figure at the top of the next page).

An executable program consists of both code (machine instructions corresponding to statements in the original C program) and data (variables in the original program). Each variable in the program occupies one or more bytes of memory;