1

Some Basic Ideas

The White Rabbit put on his spectacles. “Where shall I begin, please your Majesty?” he asked.
“Begin at the beginning”, the King said, very gravely, “and go on till you come to the end: then stop.”

Alice’s Adventures in Wonderland

There are those who will tell you that LISP is an acronym for LISt Processor and others who insist that it stands for Lots of Infuriatingly Silly Parentheses. Both camps have good arguments to back them up. Lisp is a language which deals almost exclusively in list structures, and there are a great many brackets in a typical Lisp program. Paradoxically, Lisp derives much of its power as a programming language from the fact that it is limited in this way, and, as we shall see, this philosophy leads automatically to the proliferation of similar symbols (which just happen to be brackets) which so incense Lisp’s detractors.
Lists

Perhaps we should begin, then, by studying what is meant by the term ‘list’ in the current context, and try to see why a list is such a powerful construct.

Suppose that you are writing a word processor. A simple technique for storing the text in memory would be to use an array as shown in Figure 1. There are two main objections to this rather naive format. First, an attempt to add the word ‘black’ before ‘cat’ requires that the remaining words in the text are each shuffled one cell down the array. This is OK if we’re talking about a six word sentence, as here, but if it’s the first sentence of the great American novel, there’s going to be a long wait before the extra adjective is successfully inserted. Second, there’s an implicit assumption that all words are of the same length. Of course, a word may be padded with spaces, but that will waste memory, and there’s certainly a limit to the size of word which can be held. So let’s consider an alternative arrangement, shown in Figure 2. Here, each entry consists of a word and a pointer to the next entry. The physical order of the entries no longer matters, so that additional words can be added to the text simply by creating cells for them and altering a couple of pointers (see Figure 3).

That deals neatly with the editing problem, but it still leaves us with the difficulty of handling what are effectively fixed-length words. Suppose we revise the structure of Figure 3 so that each entry consists not of a datum plus a pointer, but two pointers. Now the left-hand pointer can point to a similar structure which identifies a word and of course it can be any length, because we can signal