

---

# Preface

---

Adults worry a lot these days. They worry especially about how to make other people learn more about computers. They want to make us all “computer-literate.” “Literacy” means both reading and writing, but most books and courses about computers only tell you about writing programs. Worse, they only tell about commands and instructions and programming-language grammar rules. They hardly ever give examples. But real languages are more than words and grammar rules. There’s also literature—what people use the language for. No one ever learns a language from being told its grammar rules. We always start with stories about things that interest us. This book tells some good stories—in Logo.

The trouble is, people often try to explain computers the same ways they explain ordinary things—the way they teach arithmetic by making you learn “tables” for adding and multiplying. So they start explaining computers by telling you how to make them add two numbers. Then they tell you how to make the computer add up a lot of numbers. The trouble is, that’s boring. For one thing, most of us already hate adding up numbers. Besides, it’s not a very interesting story.

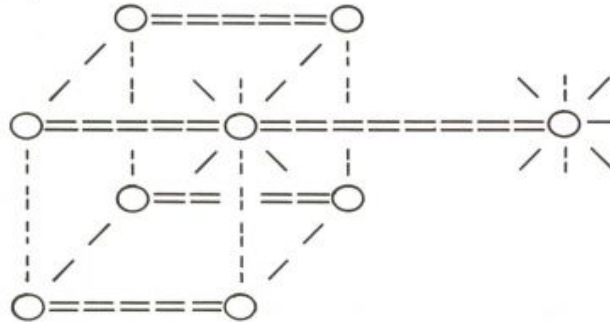
You can’t blame teachers for trying to make numbers interesting. But—let’s face it—numbers by themselves don’t have much character. In fact, that’s the real reason mathematicians like them. They find something magical about things that have no interesting qualities at all. That sounds like a paradox. Yet, when you think about it, that’s exactly why we can use numbers in so many different ways! Why is it that we get the same kind of result when we count different kinds of things—whether we’re counting flowers or trees or cars or dinosaurs? Why do we always end up the same—with a number? That’s the magic of arithmetic. It wipes away all fine details. It strips things of their character. The qualities of what you count disappear without a trace.

Programs do the opposite. They make things come to be, where nothing ever was before. Some people find a new experience in this, a feeling of freedom, a power to do anything you want. Not just a lot—but anything. I don’t mean like getting what you want by just wishing. I don’t mean like having a faster-than-light spaceship, or a time machine. I mean like giving a child enough kindergarten blocks to build a full-sized city without ever running out of them. You still have to decide what to do with the blocks. But there aren’t any outside obstacles. The only limits are within yourself.

Myself, I first had that experience before I went to school. There weren’t any Logos yet, but we had toy construction sets. One was called TinkerToy™. To build with TinkerToy you only need two kinds of parts—just sticks and spools. Spools are little wooden wheels. Each has one hole

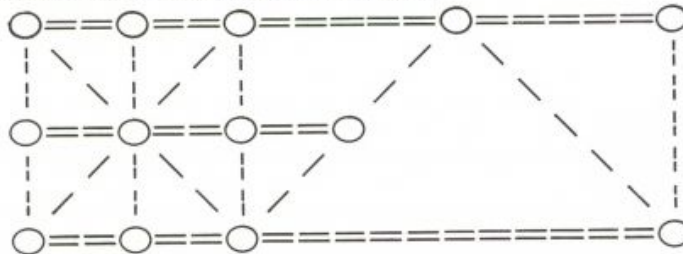
## PREFACE

through the middle and eight holes drilled into the rim. Sticks are just round sticks of various lengths, which you can push into the spool holes. They have little slits cut in their ends, which make them hold tight when they're pressed into the holes.



What's strange is that those spools and sticks are enough to make anything. Some spools are drilled with larger holes, so sticks pushed through those holes can turn. You can make towers, bridges, cars, bulldozers. Windmills. Giant animals. You can put wheels on your cars and make bearings for pulleys and gears to make them do more interesting things. You have to make the gears yourself: just stick eight sticks into a spool. They work, though not too well, and always go *click-click-click* when they turn.

The sticks are cut to several lengths. One series of lengths come in the ratios one, two, four, and eight. The other lengths are cut so that they fit across the diagonals of squares made from the first series of sticks. The amazing thing is that you can also use the first kind as diagonals for squares made with the other kind of sticks, like this:



The secret is in finding out how much can come from so few kinds of parts. Once, when still a small child, I got quite a reputation. My family was visiting somewhere and I built a TinkerToy tower in the hotel lobby. I can't recall how high it was, but it must have been very high. To me it was just making triangles and cubes, and putting them together. But the grownups were terribly impressed that anyone so small could build anything so big. And I learned something too—that some adults just didn't understand how you can build whatever you want, so long as you don't run out of sticks and spools. And only just this minute while I'm writing this, I realize what all that meant. Those adults simply weren't spool-stick-literate!

When my friend Seymour Papert first invented Logo, I had the same experience again. Logo has some things like sticks: we call them `FORWARD :LENGTH`. And Logo also has its spools: we call them `RIGHT :ANGLE`. I recognized old building friends at once. Making Logo programs is a lot like



building with construction toys—but it's even better. You can make drawings of things and structures, but you can make procedures too. You can make them use words. You can make things change their forms. And you can make them interact: just give them each procedures which can change the values of the other ones. As toys, those programs have their faults: you can't take Logo cars outside and roll them down a real hill—but, in exchange, their parts don't get loose and fall out and get lost. And the basic experience is still there: to see how simple things can interact to make more wonderful things.

Logo started many years ago; several writers of this book were children, and among the first to find new things to do with it. I'm very pleased to write this introduction now, recalling what a great adventure this has been and knowing, too, that it has just begun.

There were other good construction toys, like Erector™ sets and Meccano. They had many kinds of parts, but the basic ones are metal strips with many holes and different kinds of angle brackets. You got a million little screws and nuts to put them together with, and long steel shafts, which fit through the holes just loose enough to turn. And there are gears and pulleys to attach to the metal shafts, so you can make complicated things that really work.

When I was older, I built one of the very first modern, remote-controlled robots, using parts of a Number 10 Meccano set and ideas invented at MIT's first computer research laboratory in the 1940s. And, speaking of building computers, some of the people in this book once built a real, honest-to-goodness computer out of nothing but TinkerToy parts. A group including Danny Hillis, Brian Silverman, and Ed Hardebeck built a machine of TinkerToy parts to play the game tic-tac-toe. It actually worked and is now in a museum in Arkansas. It was made of spools and sticks. They also used some string and, since the truth must be told, they hammered in some little brass nails to keep the sticks from falling out. It took about 100 sets and was too big to fit in your room.

The golden age of construction sets came to its end in the 1960s. Most newer sets have changed to using gross, shabby, plastic parts, too bulky to make fine machinery. Meccano went out of business. That made me very sad. You can still buy Erector, but insist on the metal versions. Today the most popular construction set seems to be LEGO™—a set of little plastic bricks that snap together. LEGO, too, is like Logo—except that you only get RIGHT 90. It is probably easier for children, at first, but it spans a less interesting universe and doesn't quite give that sense of being able to build "anything." Another new construction toy is FischerTechnik™, which has good strong parts and fasteners. It is so well made that engineers can use it. But because it has so many different kinds of parts, it doesn't quite give you that Logolike sense of being able to build your own imaginary world.

About the time that building toys went out of style, so did many other things that clever kids could do. Cars got too hard to take apart—and radios, impossible. No one learned to build much any more, except to snap together useless plastic toys. And no one seemed to notice this, since sports and drugs and television crime came just in time. Perhaps computers can help bring us back.

## PREFACE

After you've built something with your construction set, you have to take it all apart again—or you won't have enough parts for the next project. With programs, you can keep them on your disk and later get them out and build them into bigger ones. This year, you might run out of memory—but that won't be a problem for *your* children, because memory will soon be very cheap. What's more, you can share your programs with your friends—and still have them yourself! No emperor of ancient times could even dream of that much wealth. Still, many adults just don't have words to talk about such things—and maybe, no procedures in their heads to help them think of them. They just do not know what to think when little kids converse about “representations” and “simulations” and “recursive procedures.” Be tolerant. Adults have enough problems of their own.

To understand what computers are, and what they do, you shouldn't listen to what people say about those “bits” and “bytes” and binary decisions. I don't mean that it isn't true. Computers are indeed mostly made of little two-way switches. But everyone who tells you this is what you need to understand them is simply wrong. It's just as true that houses can be made from sticks and stones—but that won't tell you much about architecture. It's just as true that animals are mostly made of hydrogen, carbon, oxygen, and nitrogen—but that won't tell you much about biology.

A Martian szneech once mindlinked me; it wanted to know what literature was. (It seems they're far behind the times and haven't even got to that.) I told it how we make sentences by putting words together, and words by putting letters together, and how we put bigger spaces between words so that you can tell where they start and stop. “Aha,” it said, “but what about the letters?” I explained that all you need are little dots since, if you have enough of them, you can make anything.

The next time, it called to ask what tigers were. (Apparently, they haven't even got to vertebrates.) I explained that tigers were mostly made of hydrogen and oxygen. “Aha,” it said, “I wondered why they burned so bright.”

The last time it called, it had to know about computers. I told it all about bits and binary decisions. “Aha,” it said, “I understand.”

You really need two other facts to understand what computations do. Here's the first one.

*Computer programs are societies.* Making a big computer program is putting together little programs.

To make a good program, you build a larger process out of smaller ones. I suppose you could truthfully say that sculptors make large shapes from stuck-together grains of clay. But that shows what's wrong with the bits-and-bytes approach. No sculptor or scientist or programmer ever thinks that way. An architect first thinks of shapes and forms, then walls and floors, and only last about how those will be made.

Here is the other thing most people don't know. It doesn't matter very much what you start with! Even if we start with different kinds of computers, with different kinds of parts inside—still, they mostly can be made to do the same things, when seen from the outside. You can build a windmill



with either wooden sticks or metal beams. When you look closely, each part will be quite different. But windmills will have a base, a tower, and a propeller. The same with computers:

*Any computer can be programmed to do anything that any other computer can do—or that any other kind of “society of processes” can do.*

Most people find this unbelievable. But later in this book you’ll find a hard project called “A Logo Interpreter” that shows how to make one type of Logo computer act like a different kind of Logo computer. The way to do this was first discovered by an English scientist named Alan Turing. First he asked what a computer is—and realized that the only important thing about a computer X is the set of laws that make its parts change their states. Except for that, it doesn’t matter how parts are made. Then Turing asked what programs are, and realized: a program is a certain way to fix some set of X’s states. This, then, will prearrange the way that X’s other states will later change.

Next, Turing thought, suppose you wanted a different kind of computer, Y. Then make a program for X that will make the rest of X’s states act just like Y’s. Once that’s done, X’s behavior will look exactly like Y’s—to anyone watching from outside. A programmer might say that X is “simulating Y.” Of course, you have to pay a price for this: it won’t work at all unless X has enough memory to hold a description of Y. And if X and Y are very different, then the simulated programs will run very slowly. But aside from that, Turing showed, any kind of computer can be programmed to simulate any other kind! That is why we can write special programs to make the same Logo programs run on all the different computers in the world.

In fact, every Logo system uses just such a program, to make the underlying computer act like a Logo computer. The one for Atari Logo was written by Brian Silverman and his friends. I’m sorry you can’t examine it; the trouble is that it’s not written in Logo but in a different language buried deep inside your machine. But it’s there, hiding out of sight, making your computer simulate a Logo computer. The strange thing is that Alan Turing figured out how to do such things fifty years ago, in 1936, long before computers were even invented! How could he do that? He simulated them inside his head.

There’s something “universal” about how big things don’t depend so much on what’s inside their little parts. This must be the secret of those magical experiences I had, first with those construction sets and, later, with languages like Logo. What matters is how the parts affect each other—not what they are themselves. That’s why it doesn’t matter much if one makes houses out of boards or bricks. Similarly, it probably won’t matter if aliens from outer space have bones of gold instead of bones of stone, like ours. People who don’t appreciate how simple things can grow into entire worlds are missing something important. They find it hard to understand science, because they find it hard to see how all the different things we know could be made of just a few kinds of atoms. They find it hard to understand evolution because they find it hard to see how different things like birds and bees and bears could come from boring, lifeless chemicals—by testing trillions of procedures. The trick, of course, is that it’s done by many steps, each using procedures that have been debugged already, in the same way, but on smaller scales.

**PREFACE**

Why don't our teachers tell us that computers have such glorious concerns? Because most adults still believe the only things computers do are big, fast, stupid calculations of arithmetic. Besides, our teachers have too many other things to learn and teach: how to build computers, how to make languages for programming them, and how to train programmers to use those languages. And so those dreary practicalities of billion-dollar industries crowd out our dreams and fantasies of building giant mind-machines.

*Marvin Minsky*