ADVANCE TO LOGO

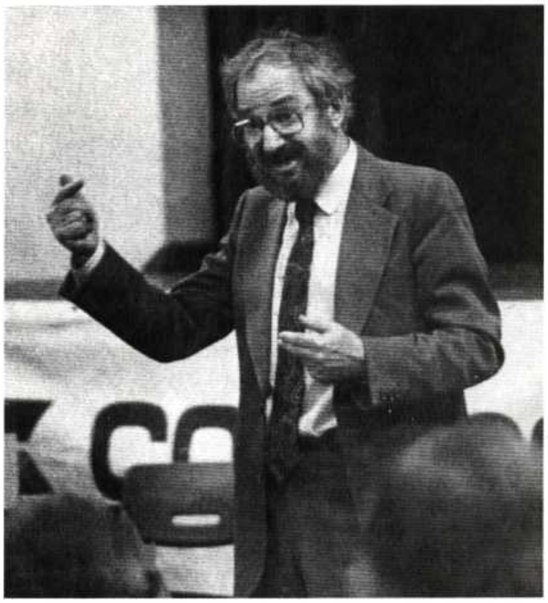
We begin a series of articles about LOGO, a programming language designed primarily with education in mind. We look at the history of its development, the basic structural philosophy behind the language and the types of users it may appeal to.

Having examined BASIC and machine code programming in detail, we begin our course on other popular computer languages with a series of articles about LOGO. You may wonder why we have chosen this language as a subject for an extended learning programme. After all, there are many other languages that perform certain functions extremely well, and home computers are rarely supplied with LOGO. Nevertheless, LOGO does offer some very attractive features to the home computer user.

First of all, LOGO is one the best introductory languages available on any computer today. Of course, if you have been programming in BASIC, you may feel little need to know about an introductory language. Yet, even for the experienced BASIC programmer, LOGO can serve as an excellent introduction to 'structured' programming and to the use of procedures instead of statements. Secondly, LOGO is available on cartridge or cassette for most home computers. In fact, one of the best versions of LOGO available is written for the Spectrum and distributed by Sinclair. Finally, LOGO is a very powerful learning system. Though not an easy language to master by any means, LOGO is one of the few programming languages that is easy to start working with.

Founding Father

Seymour Papert, the founding father of LOGO, is shown here at a conference sponsored by Commodore in 1983. Papert is now associated with LOGO Computer Systems, Inc, (LCSI), which provides LOGO programs for the Sinclair Spectrum, Atari computers, and others



Meet LOGO

LOGO has two fundamental characteristics that make it such a powerful educational language. The first is that it is interactive: when you type in a command, you immediately see the results on the screen. This means it is easy to make progress (particularly for children and beginners) because you can check yourself at every step.

The second vital feat ire is that LOGO is extendable: complete operations are handled by lists of elemental LOGO instructions. These lists are called procedures. Once a procedure has been defined as a set of particular instructions, the name of that procedure takes on the status of a new LOGO command. From then on, the entire procedure can be executed simply by typing in its name. In this way, you can actually create your own commands in addition to those 'primitives' that are an inherent part of the language.

In programming with LOGO, most people tend to be more exploratory than with other languages. Sometimes they will take a fairly strict approach and define a specific outline from the very beginning. Sometimes they will start with a core problem and write a procedure to solve it, and then build a program around that procedure. It is possible to take a flexible approach to LOGO programming because there are usually several ways to arrive at a particular result.

LOGO's origins lie in the artificial intelligence language LISP, which was invented in the early 1960s to make it easier for computers to deal with complex data structures. Its name derives from the fact that it is a 'list processing' language, which means that its basic data structure is a list, rather than a character string or numeric array, as in BASIC. LISP's essential functions manipulate the data within a list. List elements can be simple symbols or whole lists. The advantage of this approach is that non-numeric data (such as a sentence) is more easily processed in this manner.

LISP relies heavily on the principle of recursion, whereby something (usually a function or procedure) is defined in terms of itself. In the case of LISP, the item being defined is always a list. These are not accidental characteristics of LISP, but arise from its origins in computer-based investigations of natural language and human intelligence. However, the language is not an easy one to learn, and in 1968 a group of people associated with the Massachusetts Institute of Technology (MIT) set about devising a language for children based on LISP.

The charismatic leader of the group at MIT was Seymour Papert. He had previously spent a number of years studying cognitive (learning)