Randy Jones

Unlocking the System's Secrets

By PETER J. SCHUYTEN

It has been a little more than a month since I brought home my personal computer, and while I have enjoyed the off-the-shelf programs that give the computer its instructions, I wanted to learn more about the machine and at the same time take greater advantage of its abilities.

So the next step was to learn something about writing those instructions, or programming. Here's an example:

```
100 PRINT "WHITEDICE",
110 PRINT INT(6 * RND(1)) + 1
120 PRINT "REDDICE",
130 PRINT INT(6 * RND(1)) + 1
```

This is a program that simulates a pair of dice being rolled.

Some might wonder, with all the preprogrammed or "canned" software available these days for personal computers, why users would go to the trouble of learning to program at all.

But the challenge of unlocking the computer's secrets, of learning how to communicate with it in a language that

The Computer In the Home

Fourth of an occasional series.

is uniquely its own, is irresistible to some. Besides, in a very practical sense, owners of these machines are completely at the mercy of preprogrammed software when anything goes wrong.

A computer is a demanding partner. It has no patience for mistakes, abhors imprecision and becomes maddeningly balky when loaded with incorrect instructions.

On the other hand, if you take the time to learn the vocabulary and grammar governing its operations, the computer will do your bidding.

It was with this in mind that I set aside three days in which to acquaint myself with the rudiments of programming. Retreating into the den where my Apple II computer, television set and cassette tape recorder are located, I armed myself with pencils, pads of paper and a hand-held calculator.

The instruction manual reassured me that programming is a little like learning to ride a bicycle: Although it may be a bit painful at first, once learned, it is a skill not easily forgotten.

Programming is an art that loves

Continued on Page D10

Unlocking the Secrets of Your Computer

Continued From First Business Page

logic, depends on numbered coding schemes and, above all, demands immense concentration and patience. There are lists of special terms, or "reserved words," to be memorized that have a computer significance all their own, tricky computer orders to be negotiated, and editing instructions for writing and correcting programs.

There is also the matter of precedence, or the order in which the computer performs mathematical operations. (Minus signs indicating negative numbers are executed first, followed by exponentiation, multiplication and division, addition and subtraction operations, all carried out from left to right, unless otherwise instructed.)

The hours pass quickly as you move from one sample program to the next, until you are able to produce clicking, ticking and tocking sounds on the computer and plenty of splashy color graphics.

At other times, however, you begin to wonder why all this is being learned.

The key to all programming, you discover, is the stored, or deferred, execution statement, which is an instruction that you have stored in the computer's memory for later use. Type enough of these in the computer's memory — written, of course, in the correct format and sequence — and you have created a program.

As you delve deeper, you learn how to clear the memory of these instructions (typing the command "NEW"), as opposed to simply clearing the screen ("HOME"), while the command "RUN" orders the computer to carry out these instructions.

Next comes "looping," which directs the computer to return to a previous instruction, or line number, which it will then repeat endlessly until told to do otherwise.

Progress Can be Slow

Everyone is said to have at least one good program in him. But your progress can be agonizingly slow when encountering such head-scratching statements as: "The numbers that the function uses are called its arguments and are always put in parenthesis after the function name. PDL is a function that has one argument."

Computers can distinguish between truth and falsehood in mathematical statements, also called assertions. Ask it, for example, if six is greater than two and it will respond with a "1," meaning the statement is true, while the reverse produces a "0," meaning it is false. Beyond that, however, the computer cannot really distinguish degrees or shadings of truth, and it has absolutely no imagination. In its ordered mind, the assertion "Not 1" is "0" and "Not 0" is "1," and everything else is simply a variation on these two assertions.

In cases where the computer cannot distinguish between truth and falsehood, that is, when you have entered something via the keyboard that confounds its sense of syntax, it will unfailingly respond with the expression "?SYNTAX ERROR," meaning it did not understand.

After spending an hour or more creating a program only to be rewarded with a "?SYNTAX ERROR" message on the screen each time you attempt to run it, you begin to be overcome with frustration. By contrast, there is an exhilaration when the computer executes your program instructions without a hitch.

More than anything else, you discover that programming is like learning to cook. At first you slavishly follow the recipes, or sample programs contained in the manual, painstakingly keying them into the computer, step by step. But later, as you gain confidence, you begin to improvise here and there, until finally you are creating programs of your own.

A Few Warnings

A word of warning: A simple but well-crafted program — one, for example, that fills the screen with rapidly changing color patterns, or adds the "bounce" sound to a computer-generated ball — may involve hours of work at first. But in time, it may become like child's play.

Another warning: Don't expect others to be overwhelmed by a virtuoso computer performance. Like a baby's first halting steps, programming triumphs are of interest only to the parent.

The first program I was able to create without prompting from the manual involved instructing the computer to draw an endless series of random lines of varying lengths and colors across the screen. That in turn was followed by programs that generated intricate moiré patterns in which colors continuously alternated, a bouncing ball that emitted a different sound each time it rebounded off the walls of the display screen, and finally a program that transformed the display screen into a sketch pad, in which the computer's game paddles — which come with the computer and control the action on the screen — could be used to draw lines. In each case my friends were not terribly impressed.

Some might say that what I have been doing is devising a series of ingenious but rather useless computer tricks, and that the serious business of programming lies beyond by these simple skills.

To a degree that is correct. But cleverness aside, these elementary programs, and the knowledge that underlies them, is part of the process of understanding how the computer "thinks," and that is the central ingredient needed by anyone seeking to take the next step toward turning the computer into a useful tool. Besides, there are three programming manuals to conquer.