LANGUAGES/JOHN R. WILSCHKE

Grandfather FORTRAN

Known chiefly as a scientific programming language, FORTRAN has held its own in the computer work place for 26 years.

If you have been reading *PC*, you have probably read about the newer programming languages now available for the PC, such as Ada, APL, C, FORTH, and Pascal. At the same time, one of the major languages in computing, FORTRAN, has been receiving relatively little attention. This is due partly to the poor quality of the original IBM FORTRAN compiler for the PC and partly to misconceptions about this "grandfather" of programming languages in the microcomputer world. Actually, FORTRAN can be a very powerful microcomputer language that should be seriously considered by those wishing to move up from BASIC. I will explore some of the reasons you might choose FORTRAN. But first, FORTRAN's history.

FORTRAN, which derives its name from FORmula TRANslation, is 26 years old. It was developed by IBM in the mid-50s and the first version was released in April 1957. FORTRAN I was followed by FORTRAN II in 1958 and FORTRAN IV in 1962. Although originally developed for the IBM 704 machine, the language was so popular that it was soon adapted to most of the computers of the time. FORTRAN was the first computer language standardized by the American Standards Association, which is now called American National Standards Institute (ANSI). The standard produced FORTRAN 66 (ANS X3.9-1966), which roughly matched IBM's FORTRAN IV. More than 10 years later, a new standard, FORTRAN 77 (ANSI X3, 1978), provided FORTRAN with enhanced features that overcame previous difficulties with text-type data. FORTRAN's input/output (I/O) capabilities were also adapted for use on interactive computer terminals. IBM supports this standard in its version VS FORTRAN for use on the mainframes. Although more code has probably been written in FORTRAN than in any other language, it still suffers from its reputation as only a scientific language. However, its flexibility has allowed it to be used for many general computing applications. It was used to create the Statistical Package for the Social Sciences programs (SPSS), which is used by many university researchers to analyze statistics for insights into social behavior. Its use in computer design packages has been an aid not only to architecture and engineering, but also to industrial and fine arts graphic creation. It has been used for computerized machine control in the manufacturing of automobiles and aerospace vehicles. The original game, *Adventure*, was written in FORTRAN at the Massachusetts Institute of Technology. And although the business world still relies heavily on COBOL, FORTRAN has been useful in programming the higher mathematics of business strategic planning and risk analysis. All branches of the United States military currently recognize FORTRAN as a suitable language for military programming.

FORTRAN is still growing. In the next few years a version—possibly called FORTRAN 8—will be written to include many of the advanced features that now are in Pascal and Ada without losing commonality with the existing FORTRAN standards. Thus, far from being left in the dust by the newer languages, FORTRAN will remain a major computing language in the decades to come.

Unpopular with Micros

Yet FORTRAN has not been popular in the micro world. I believe this is due to several factors. First, it is a compiled language. Compiled languages, in contrast to interpreted languages, give up debugging convenience for gains in execution speed. It may take 10 minutes or more and a lot of floppy disk handling to compile a program on the PC. However, once compiled, this program will run 10 to 100 times faster, depending on the length and complexity of the original program. The execution of interpreted languages is slow, because an interpreter must translate each line of source code into machine language as the program runs. In compiled languages the translation is done once at the time of compilation and produces a separate "machine language" program that will be used for execution. Also, interpreted languages usually limit the size of the program and the amount of data that can be handled at any one time. Because program development of compiled languages requires considerable time and effort, there was little incentive to use them as long as interpreted programs were small enough that execution speed wasn't a problem. As the micro

world progresses into bigger and more complex programs, the use of compiled languages, including FORTRAN, will become more popular.

Another reason why FORTRAN never attained much status in the micro world is

All branches of the United States military currently recognize FORTRAN as a suitable language for military programming.

because it is defined by reasonably strict standards, and consequently, it was not easy to produce an effective version of FORTRAN for the early 8-bit micros. The mathematical and I/O library routines required large amounts of memory and left very little for the actual FORTRAN program. On the PC, at least 128K is required by all the known compilers to compile a FORTRAN program and fully support the language's capabilities.

The early users of micros seemed to be either very experienced computer buffs or new computer converts. The experts liked the newer, more powerful and structured languages like Pascal or the efficiency of Assembly language; the converts liked the easy-to-learn BASIC or application packages like *VisiCalc*. There wasn't a place for FORTRAN. As the field of microcomputing moves from the experimentation phase into the mainstream of the computing work force, a new and large group of professional programmers will be using PCs to do the work they once did on the mainframes. It's likely that these programmers will use major languages such as FORTRAN for the same reasons they used such languages on the mainframes.

For a new computer programmer who wants to progress from the interpretive BASIC into the professional languages, the number of choices is confusing. Languages such as Pascal or Ada have a number of advantages, but FORTRAN has its merits, too.

First, it is easy to learn; FORTRAN provides its basic capabilities with as few instructions as possible (KEYwords or Reserved words). If producing your first program takes a semester's course, FORTRAN is too complicated a language for your current programming level. You can save yourself a lot of headaches by setting your sights a bit lower. However, for a BASIC programmer, little effort is required to learn FORTRAN; of all the programming languages, it comes closest to BASIC both in the form of the code and in the types of variables used.

A major difference between FORTRAN and BASIC is that BASIC has line numbers, while FORTRAN has none. FORTRAN statements are executed according to the order in which they are placed in the lines of code—one statement to a line, beginning in column 7. (This is not entirely true for optimizing compilers, but you can ignore this technicality.) In FORTRAN there are numbers that precede some statements, but they are called LABELS. Their function is to indicate the destination of logical branches like GOTO statements, to define a type of PRINT USING statement called FORMATs, and to locate the end of a loop. Labels appear only where they are needed and do not necessarily appear in numerical order. (This, however, is a very good programming practice which aids program readability.)

The data types of BASIC are available in FORTRAN. Integers in FORTRAN are assumed to begin with the letters *I,J,K,L,M,N* rather than ending in a percent sign (%) as in BASIC's. In FORTRAN 77, BASIC's string variables are called character variables and are slightly less convenient to use. FORTRAN can also have Logical, or Boolean variables, which can be assigned the values of either true or false. These variables are useful in making complex logical decisions in program control. In addition, some versions also support Complex variables, which are needed for the mathematics of imaginary numbers.

The BASIC programmer should have little trouble understanding a FORTRAN program after only a few minutes of instruction, and learning to write FORTRAN programs wouldn't take much longer.

FORTRAN is easy to use, which will help you keep your programming neat and understandable. Programming should emphasize, rather than disguise, the logical flow of the program. If you cannot find the error and are convinced "it just can't be there," perhaps the complexity of the language is masking what the computer is actually doing. Any language should help the programmer to understand any other program in that language.

This has been one of the major complaints professional programmers have about BASIC and, to a lesser extent, about FORTRAN. FORTRAN's developers gave little thought to a programming language theory. Hence, FORTRAN can be abused. There is a dangerous possibility of assigning different variable names to the same storage location through the use of EQUIVALENCE and COMMON statements. With indiscriminant use of the GOTO branches, one can send a program into logical convolutions so complex that no one can understand the program's purpose. Not only are these programs hard to modify, they are prone to hidden errors of which even the programmer is unaware.

While FORTRAN's design permits poor programming, it also provides the capability to program in an understandable and structured way. The FORTRAN 77 standard restricts the branching into loops and provides the IF THEN ELSE (blocks IFs) structure, allowing for structured programming in much the same way that Pascal and Ada do. In a future issue of *PC*, I'll discuss more of FORTRAN's programming features and then take a look at some of the new FORTRAN compilers now available for the PC. ■