FORTRAN: Not Just For Scientists

Though FORTRAN rivals Pascal and Ada in structure, the language still allows for the writing of quick and dirty programs.

In the article, "Grandfather FORTRAN," which appeared in this column in the last issue of *PC,* I talked about FORTRAN's entry into the microcomputer world and discussed some of its programming features. As you recall, this language allows for structured programming, just as Pascal and Ada do.

While programming experts would prefer to have a language require strict programming practices, such structured languages as Pascal, Ada, and PL/I tend to be too cumbersome for small programs. At the same time, these constraints are welcome and needed when large complicated programs are created, debugged, and later modified. FORTRAN allows both types of programming and requires the programmer to know when to use each. In addition, the future version of FORTRAN will provide further structural capabilities that will enhance its ability to provide clear and precise code. FORTRAN will then rival Pascal and Ada in structure, but it will still allow "quick and dirty" programs to be written.

This is the second of two articles on the programming language FORTRAN.

Another ease-of-programming feature offered by FORTRAN is its ability to allow a program to be broken into separately compiled subroutines. All of the information passed between the main program and a subroutine, or between two subroutines, is tightly controlled through Call statement arguments (for example, CALL TIME (HRS, MIN, SEC)) or Common Blocks (for example, COMMON /BLOCK1/ HRS, MIN, SEC).

Unlike BASIC, this is the only way variables are shared between calling and called routines. This allows a programmer to debug his program in easily understandable blocks and to assemble those blocks into the final program with the linker. A change in one block does not require that the whole program be recompiled, which prevents you from wasting time. The FORTRAN programmer can then create a "library" of already debugged and compiled subroutines to be used in new programs.

FORTRAN's popularity is well established. You rarely hear the argument that a programming language is good because it is popular, but popularity should be a factor. A good computer program, like a good novel, is more valuable if it can be used and understood in more than one place. Rarely is a program transportable among different computers if the language used is not supported on both computers. Even when the same languages are supported, if they are not standardized versions, one must be wary of differences. One of FORTRAN's outstanding features is that it is so standardized that it is available for many different computers—it is one of the most transportable languages available.

FORTRAN subroutine libraries that contain proven solutions to many problems exist in the public domain and are for sale. IMSL, Inc. (NBC Bldg., 6th Floor, 7500 Bellaire Blvd., Houston, TX 77036) is planning to offer a subset of its over 500 FORTRAN subroutines for use on microcomputers. Also, at your local college library, you'll find a huge pool of FORTRAN programs and solutions for almost any field of computing.

FORTRAN is an efficient and powerful language. There seem to be some standard features necessary in any general programming language. The ability to manipulate high-precision numbers quickly will provide the speed and accuracy required for many business and scientific/engineering applications. The ability to perform string/character operations makes word processing and report generating programs practical. The ability to program in separate modules for later combination with more complex programs greatly increases the power of the programmer to organize and develop sophisticated applications. A language that fails to provide these requirements as standard has limited use. The better implementations of FORTRAN contain all of these features.

But why not use the popular Pascal? While it is an International Standards Organization (ISO) language, the standard itself is not very complete. Standard Pascal does not support such important features as exponentiation, double-precision real numbers, complex arithmetic, or modular programming. Although IBM Pascal has exponentiation and modular programming, there is no assurance the IBM enhancements will conform to a future Pascal standard. The "power" that FORTRAN lacks, when compared to Pascal, is the ability to directly access the memory or operating system. If the POKE- and PEEK-type commands are important to you, FORTRAN may not be for you. However, the ability of a FORTRAN program to incorporate subroutines written in Assembly language, or even in languages such as Pascal, provides an alternate means for those who wish to access the most primitive parts of the computer.

FORTRAN Compilers

The ability to use any language depends upon the availability of good compilers/interpreters. New FORTRAN compilers are now appearing for the PC, and each one seems to be better than the last. While so far none has the full power of the FORTRAN 77 standard, before long one that does should appear. The IBM PC, with its large addressable memory space and the 8087 numeric coprocessor, is one of the first micros that could fully support such a compiler. Four of the available computers are looked at here.

IBM FORTRAN Version 1.0 is an unfortunate choice. At last count there were 45 pages of patches for known errors in this compiler. If you own IBM's FORTRAN 1.0, you can get a patched update from your dealer.

This compiler lacks double-precision real numbers and uses a real-number format that is incompatible with the 8087 numeric coprocessor. While it supports a subset of the FORTRAN 77 standard, it has some serious omissions, such as the lack of list-directed I/O for easy communications from the keyboard to the program. It also poorly documents the compiler defaults. Many users are unaware that the compiler generates up to two-thirds more code than necessary. It is hard to understand why the originator of FORTRAN would put out such a poor product.

The FORTRAN for the UCSD P-Code system operates under the UCSD system, which is not the *defacto* standard for the IBM PC. UCSD FORTRAN incurs an efficiency penalty by compiling only to a P-code level. This P-code must then be interpreted at run time, just as it is in interpretive BASIC. However, if you use the UCSD system, this is the compiler for you.

Supersoft FORTRAN IV Extended (Version 1.04) could be the most efficient FORTRAN compiler available for the IBM PC. In tests, its programs run two to three times faster (when an 8087 is not used), than those of the IBM or Microsoft versions. The compiler has double-precision real numbers and complex numbers. (Complex numbers are very powerful tools for solving problems in areas such as electrical engineering.) It uses the IEEE real-number format, and an optional package can be purchased to access the 8087 coprocessor. Also provided is an assortment of useful subroutines, which allow a FORTRAN program to access the operating system and maintain control of the screen.

Unfortunately, like the IBM BASIC and Pascal compilers, the Supersoft FORTRAN IV compiler limits the amount of data space to 64K. This includes all common data (shared between subroutines), local variables (only used inside a subroutine), format definition statements (for input and output), the stack (for transfer of data between subroutines), and the heap (file control blocks). Even for moderate-sized programs, 64K can be a limitation. In addition, the commonly used 4-byte integer is not allowed. I was told by Dale Jurich, the author of the Supersoft FORTRAN IV Compiler, that future revisions will correct these problems.

But probably the most serious problem is that it is not close to a standard FORTRAN. While it calls itself a FORTRAN IV(66), it has included some but not all of the FORTRAN 77 enhancements. FORTRAN 77 features, such as character data, are manipulated with subroutine calls rather than with the standard operators. The interactive list-directed I/O and error handling are also non-77 standard.

Microsoft's FORTRAN 77 Version 3.1 has a language structure very similar to the one offered by the original IBM FORTRAN compiler. It even retains many of the same metacommands for control of the compiler options. If you have developed a program for the IBM version, it should compile almost immediately with MS-FORTRAN 3.1. What makes this one different from the IBM version is that this one works. Microsoft has not only fixed the errors of the IBM FORTRAN compiler, but provided much more.

It now allows double-precision real numbers, uses the IEEE floating-point number format, and allows access to the 8087 coprocessor. A large program took 1.5 hours to run using code produced by the Microsoft compiler; when the same program was run on a PC with an 8087, the running time was 6 minutes!

Microsoft's compiler still seems to produce inefficient code compared to the Supersoft version, at least without use of the 8087. It also does not provide the FORTRAN 77 character operators. This puts a penalty on those programs that do a lot of text manipulation. And the FORTRAN manual needs better explanations of the error messages (as do most other manuals). However, this is a very workable compiler. Perhaps, when the microcomputer world discovers it, Microsoft will be encouraged to provide further improvements such as the character operators.

These days, FORTRAN is no longer just a language for scientists. It is easy to learn and should be attractive to nonprofessionals moving up from BASIC. If you choose FORTRAN, you will be using an efficient and powerful language that will improve in the future without sacrificing the hard-won lessons of the past. ■