Computers in Family Practice

Editor: Roger A. Rosenblatt, MD, MPH

BASIC or Pascal: Which Is the Language for the Generalist?

Peter Mullins, MSc
Auckland, New Zealand

Mr. Mullins is a biostatistician and software author at the University of Auckland Medical School in New Zealand. Microcomputers are in wide use in the medical school environment in both New Zealand and Australia, and this paper addresses the important question of which computer language is most suitable for the physician interested in mastering the microcomputer.

Mr. Mullins and his colleagues have developed an elegant statistical package for the APPLE and IBM microcomputers that is particularly suited to rapid and painless analysis of small data sets.

Roger A. Rosenblatt—Editor

The past eight years have seen the rapid evolution of new computers, new computer languages, and a multitude of new computer applications. The notion of a personal computer first arose about 1975: the computer as a general-purpose tool for the nonspecialist. This article deals with the choice of a programming language for the generalist with little experience with computers.

One of the great generalists of the last century was Sir Richard Burton, the explorer, poet, and linguist, who at his peak had some 35 languages at his command and who claimed to be able to absorb most of the grammar of a new language within three months. What would he have made of BASIC or Pascal? The number of languages available to the microcomputer user is enormous and might even have challenged the abilities of a linguist of the caliber of Burton. This discussion will be confined to BASIC and Pascal, probably the two most popular computer languages and the two most suitable for the beginner.

To determine which of these languages is the language of choice, one must first consider the needs of the generalist. Most physicians use a microcomputer for financial management, some word processing, data management, and entertainment. Most of these needs, of course, are best met by the purchase of off-the-shelf software packages, and the language in which the package is written is largely irrelevant. If this were to be the limit of involvement and interest, there would be no need to learn any computer language.

For those who wish to use the microcomputer as more than a fixed-purpose tool, however, it is desirable to be conversant with at least one of the major programming languages. BASIC and Pascal have emerged as the leading contenders for versatile and freely available general-purpose languages suitable for beginners. There are major differences between them, however, and fluency in one is of little specific use in the other. Since learning a computer language is akin to learning a human language, such as German or French, the novice should make a considered decision in his or her selection. Both BASIC and Pascal allow the computer user to design and implement programs for a wide variety of tasks, and both can be used on virtually all microcomputers.

Since the advent of the personal computer, the language BASIC (Beginners' All-purpose Symbolic Instruction Code) has become almost ubiquitous—rare indeed is the computer that is not equipped with a version of BASIC. In the case of microcomputers, this language system is provided "free" with the machine, perhaps one of the reasons for its ubiquity! In fact, BASIC may now quite reasonably lay claim to the property of being the most widespread computer language in the world.

From the Department of Community Health, Auckland School of Medicine, Auckland, New Zealand. Requests for reprints should be addressed to Mr. Peter Mullins, Department of Community Health, Auckland School of Medicine, Private Bag, Auckland, New Zealand.

On the other hand, BASIC may also fairly be described as the most criticized computer language in the world. Reviled the world over by departments of computer science in universities, it is nevertheless still widely used in the production of commercial and scientific software and is widely taught, mainly because it is extraordinarily easy to learn, consisting of a small set of possible commands that are very "English-like" and have a very simple grammar.

A second fundamental feature that has made BASIC addictively easy to use is its interpretive mode of use: BASIC throws the programmer's mistakes back at him immediately rather than producing error messages within a printout for later perusal. This quality makes it very easy to write and test small programs quickly: debugging small BASIC programs can be astonishingly quick. When a BASIC program runs in the computer, what is actually executed is segments of machine code, but the conversion to machine code takes place line by line. There are two major side effects of this interpretive mode: routines run quite slowly, and some of the address space of the computer is occupied by code that is never used. This space is at a premium in a small machine, so that data sets are unnecessarily limited in size.

Instructions in BASIC code are written as a simple sequence of instruction lines. Each line must be numbered, and instructions will be carried out in the order of the numbering, not the order in which the instructions were entered. Execution of the program can depart from this linear sequence as a result of a decision or iterative process, by reference to these line numbers in one or other of the available control statements. The three types of control statement in BASIC are an unconditional branching statement, the GOTO statement; a conditional branching statement, the IF . . . THEN statement (some BASICs allow IF . . . THEN . . . ELSE statements); and the FOR . . . NEXT loop, for iterative execution of program segments. These statements are sufficient to give the programmer great power in manipulating data, but they also offer enormous opportunities for the creation of confusing and confused code. This lack of structure is probably BASIC's greatest drawback.

BASIC's main rival, in numerical and philosophical terms, is Pascal, named by its creator, Niklaus Wirth, for the 18th-century mathematician and philosopher Blaise Pascal. It is a rival in numerical terms in that almost every personal computer on the market today is equipped, or can be equipped, with a Pascal compiler. Here, in fact, is the greatest difference between the two languages: BASIC is interpreted, while Pascal is designed to be implemented as a compiled language. A programmer using Pascal works roughly as follows:

1. Write the program, using the syntax and grammar specified by the Pascal language.

2. Compile the Pascal program. This step actually entails turning the Pascal statements into machine language instructions, which is done with a special program called a compiler. The result of compiling a program is a new program, often called the "object file," consisting of the equivalent machine language code.

3. Link the resulting "object file" to any system programs it may use. (This step is automatic on many Pascal systems, but it nevertheless happens as part of the process and takes time.)

4. Run the program. This step is then followed by the "debugging process," which of course involves rewriting the text file, recompiling, and linking, and so on.

This process can be very tedious, particularly with large programs on small machines, since most of this processing is disk based, and the disk drive is the slowest part of a microcomputer system. Among others, these are reasons why so many people prefer to program in BASIC. On the other hand, the code resulting from a compiler runs much faster, sometimes on the order of ten times as fast as the comparable BASIC code.

The other great advantage of using Pascal as a programming language is its obvious structure: a well-written Pascal program can usually be understood by novices with very little difficulty. This structure arises from a reductionist approach to the programming problem: the overall task (process a file of data, for example) has been broken down, or reduced, to a number of conceptually easier tasks, such as "get the next record," "add these results," etc. The structural approach is also useful in writing BASIC programs, but the effect is not so obvious in the final result.

Figure 1 illustrates the differences between the two languages by showing the same program written in BASIC and in Pascal. The program is a trivially simple one; directing the computer to add together two numbers and print the result. The differences in style between the two programs are obvious and should give some appreciation of the flavor of programming in each of the languages.

The major distinction between the two lan-

BASIC

```
10    GOSUB 1000
20    GOSUB 2000
30    END
1000  A=1
1010  B=2
1020  C=A+B
1030  RETURN
2000  PRINT A" + "B" = "C
2010  RETURN
```

Pascal

```
PROGRAM ADDITION;
VAR A,B,C: REAL;

PROCEDURE ADD;
BEGIN
   A :=1;
   B :=2;
   C :=A + B
END;

PROCEDURE PRINTRESULT;
BEGIN
   WRITELN(A, ' + ',B,' = ',C)
END;

BEGIN
   ADD;
   PRINTRESULT
END.
```

Figure 1. Calculating the sum of two numbers, comparing BASIC and Pascal

guages is evident even from this oversimplified example. BASIC is a sequential language. The flow of the program is determined by the line numbers that precede each instruction. Although subroutines are frequently invoked in BASIC programs—indicated by the command GOSUB in this example—they are embedded in the larger program and can be difficult to find and to follow. On the other hand, BASIC is quick and parsimonious; it is possible to accomplish complex tasks with simple and sparse language.

Pascal, by contrast, is a verbose language. This verbosity may require more effort both to learn and to write, but the resulting program instructions tend to be self-explanatory or, in the jargon of the trade, self-documenting. As can be seen from this example, each component of the Pascal program is broken down into a self-contained procedure, and the program works by invoking these procedures. It is very easy to build up a complex program by assembling the building blocks together into a coherent whole, and such programs are easily expanded, modified, or moved from one microcomputer environment to the next.

For these reasons, Pascal offers many advantages as a systems development language because of its portability, its procedural structure, and its consequent ease of maintenance. In fact, the Apple Lisa uses Pascal as its system language. To be most effective, Pascal needs large memory, good file handling, and a sophisticated operating system. Most manufacturers now offer a version of Pascal on their machines, but BASIC is still the usual built-in offering.

Another advantage of Pascal as a programming language is that it is currently at the peak of its popularity. It is not likely to disappear over the next few years, particularly since so many computer science departments see it as the language of choice for teaching programming skills, but it may wane a little in popularity as more and more memory becomes available on microcomputer systems, and even more sophisticated languages become popular.

In conclusion, then, BASIC is far easier to learn and easier to use than Pascal, while Pascal gives code that executes much faster, that is self-documenting, and that is easier to read and maintain than BASIC. The language of choice depends largely on the type of use envisaged: the programmer who wants to develop software systems that are to be robust and last perhaps for years with some regular maintenance will find better service from the elegance and speed of Pascal. The programmer who is going to write many small programs for private use only will probably prefer the ease of developing BASIC programs. Mastering either language will enhance the generalist's understanding of computers and computing and expand the versatility and utility of the home computer.