FEATURE ARTICLES

Microcomputer Software. 1. After FORTRAN†

JOHN C. MARSHALL

Department of Chemistry, St. Olaf College, Northfeld, Minnesota 55057

Received June 12, 1983

The wide use of inexpensive microcomputers has been accompanied by the ready availability of a large variety of new computer languages. The most popular of the new languages is the strongly typed language PASCAL. Is PASCAL going to replace FORTRAN as the language of choice among chemists?

The explosive developments in microcomputer hardware have been accompanied by a less spectacular but highly significant changes in the variety and nature of the software available to the scientist. The present microcomputer scene, both in software and hardware, is very confusing. Just trying to remain informed about the things that are happening is a full time job; trying to understand all these things is simply impossible. This is the first of what I am sure the editor hopes will be timely (and on time!) articles on aspects of microcomputer software. These articles will focus on topics generally related to software facilities available on microcomputers. I hope the topics selected are of interest and the articles useful. I would be very pleased to receive comments about subjects discussed and subjects that should be discussed. The focus of this article will be on the new high-level languages, using the language PASCAL as the archetypical example.

The wide availability of powerful and inexpensive microcomputers makes it possible for the scientist to take advantage of software developments in a way and on a time frame that was not previously possible. In the past, almost all of the computing at any institution was supported by (at the mercy of?) a central computer facility. New software support from such a facility is normally a painfully slow and very complex process. First the computer center staff must be convinced that the software requested is of sufficient general interest to warrant support. If that hurdle can be surmounted, then the selected package must be adapted to the local operating system, which as a result of constant tinkering (did they ask the users if they could do all that tinkering?) is like no other in the world. Finally, the documentation must be revised to "meet local standards", which will guarantee that it is exactly as incomprehensible as all the rest. Contrast this with going down to your local computer store during your lunch hour and purchasing a complete new language processor and the associated documentation for the microcomputer in your office.

Perhaps as significant as the ready availability of a wide variety of diverse types of software is the fact that such software must compete in a generally nontechnical arena. Not only does that make the pricing structure of such software very reasonable but it also forces a very major change in the standards and tenor of the documentation. This addresses a long-standing difficulty faced by scientists for whom the use of the computer is essential for their research. While most scientists using major computer installations have come to some accommodation with the likes of the System 360/370 Job Control Language (JCL), I am sure that most would agree that any computer system with this complex and obscurely documented a user interface would never have become a major commercial success in a truly competitive environment. This is not meant to imply that all commercially available microcomputer software packages are well documented because they are not. The point is that many are, and those that are not will not survive. Anyone familiar with the microcomputer software market could point out that one very notable exception to that last statement is the most widely used of all microcomputer operating sytems. However, even this historic problem seems to be yielding to the pressure of the market.

Figure 1. Copyright retained by J. C. Marshall.

Figure 2. Copyright retained by J. C. Marshall.

The easy availability of a dazzling array of software begs yet another question. Which of the currently available high-level languages are of value? To become proficient in any high-level language requires a very great investment of time. I doubt that there are many scientists that would have the time, energy, and emotional stability to join a "language of the month club". For reasons both good and bad, chemists have historically used FORTRAN (Figure 1) to the virtual exclusion of all other high-level languages in their research. Whatever is good or bad about FORTRAN, one thing is clear,

0095-2338/83/1623-0091$01.50/0

it will survive and remain useful for the indefinite future. The significant question would seem to be, does the limited and strongy numerical philosophy of FORTRAN limit the scope of potential applications of the computer to chemistry. In the opinion of this writer the answer is clearly that it does.

The artificial mapping of nonnumerical concepts into the digital domain to accommodate the FORTRAN language makes many chemical problems much harder to solve and considerably more abstract than they should be.

On the assumption that the chemist should be searching for the heir apparent to FORTRAN, what are the possibilities? Certanly BASIC (Figure 2) has assumed a very important role in recent years. This language, originally consisting of seven fundamental language elements and designed to combat computer anxiety among undergraduate students, has many very attractive features. Among them are humane and flexible syntax, facile text handling, and instant gratification. The possible future application of this language to major computational problems should no be ruled out. The language is now in what could be termed its baroque phase.[1] Some forms of structured BASIC are already available which provide powerful extensions to the language that may make it a contender, in compiled form, for large computationally intense applications.

PASCAL is the glamour language of the moment. This is quite a significant when one considers that ALGOL, a very similar language, never gained any real measure of popularity. Computer scientists generally consider ALGOL as one of the most significant language developments ever made, and the present epidemic popularity of PASCAL may prove them right. PASCAL is not a perfect language. Indeed, it has been soundly criticized by Kernighan.[2] In its "standard" form it is nearly useless. This is not surprising because PASCAL was designed as a "play" language to teach programming rather than to be intrinsically useful. However, the language has been extended and improved in the past several years to the point that it deserves the careful consideration of all present users of FORTRAN. Unfortunately, getting started with PASCAL is a problem. The language absolutely reeks with pedantry. It is, in many ways, more syntactically rigid than FORTRAN, and to the jaded appetite of the BASIC programmer, it seems intolerable. Unlike FORTRAN, PASCAL is a strongly typed language, and the type correspondence checks are carried to absurd limits. It is unfortunate that many dedicated FORTRAN programmers never get over being mad at the language long enough to examine its real strengths. Another point worthy of note is the fact that the flexibility of the language is such that a really comprehensive manual is not possible. I must confess that my initial opinion of PASCAL was the same as my initial opinion of UNIX, the rage operating system of the moment. Initially I considered both to be long on style points and short on user convenience. After considerable experience with both my opinion of PASCAL changed. PASCAL allows a flexibility and clarity of expression that simply cannot be approached in FORTRAN. The difference between the two languages could perhaps best be summarized by saying that FORTRAN requires you to shape the problem to fit the language whereas in PASCAL you can shape the language to fit the problem. For example, note the simple program segment shown below.[3]

```
VAR
  CHARACTER : CHAR;
  LETTER    : ARRAY (.'A'..'Z'.) OF INTEGER;
BEGIN
  REPEAT
    READ(CHARACTER);
      IF CHARACTER IN (.'A'..'Z'.) THEN
      LETTER(.CHARACTER.):=LETTER(.
    CHARACTER.)+1;
  UNTIL EOL(INPUT); END.
```

The problem solved is simply to total the frequency of capital letters read from a line characters. The point is this. Contrast in your mind the way this simple problem would be solved in FORTRAN, a language with legendary disdain for text of any kind. Even BASIC, renowned for its ability to handle text the solution of this problems, would require a bit of fussing about. In any case, I contend that this simple example illustrates two very significant points. First, the problem has been clearly stated in terms of the problem to be solved. Second, the IN statement is just one of many powerful constructs in PASCAL that give the user full-dress Boolean logic, with computer-manipulated Venn diagrams being the result.

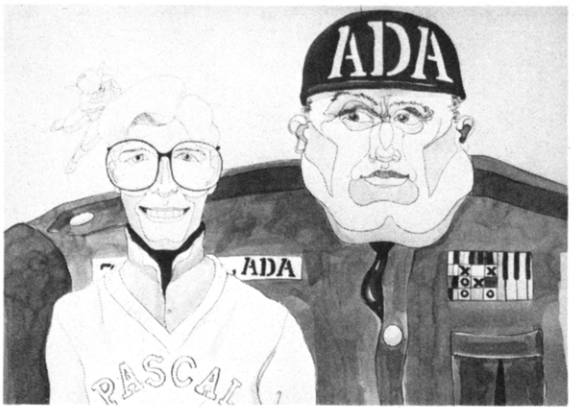

Figure 3. Copyright retained by J. C. Marshall.

So, is PASCAL or the massive superset of PASCAL, ADA (figure 3), going to be the answer? Perhaps the final answer will be found in MODULA.[4] Probably none of the above will be the case. However, the PASCAL philosophy does seem to point in the right direction. This language has gone far beyond FORTRAN in terms of its ability to allow the user to describe a problem in terms of the problem. This addresses one of the most difficult tasks associated with writing and debugging a FORTRAN program, namely, trying to remember what the problem really was when you look at the code you use to describe it.

Let me attempt to summarize my remarks in the following way. A scientific paper is a description of a problem and a solution to that problem. A correctly coded computer program is, in fact, the same thing. The reason that the two are presently so very different is that we have inferior computer languages. Most microcomputers support a wide array of high-level languages. Most chemists face a wide array of divergent problems, only a small fraction of which are strictly numerical. With all of this in mind, does it make sense to ignore the languages "After FORTRAN"?

REFERENCES AND NOTES

† Derived, in part, from a paper of the same name presented at the Seattle Meeting of the American Chemical Society, March 21, 1983.

(1) Marcellus, D. H. "A Fearless Look at Micro Changes". *Microcomputing* **1983**, *7* (Mar), 114.
(2) Kernighan, B. "Why Pascal is Not My Favorite Language". Computing Science Technical Report No. 100; Bell Laboratories: July 18, 1981.
(3) From example given by: Anderson, R. W. "From Basic to Pascal"; TAB Books, Inc.: Blue Ridge Summit, PA, 1982; p 9.
(4) McCormack, J.; Gleaves, R. "Modula-2". *Byte* **1983**, *8* (Apr), 385.