Winter 1-1-1977

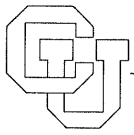
A Comparison of PASCAL and FORTRAN as Introductory Programming Languages ; CU-CS-101-77

Gary J. Nutt
University of Colorado Boulder

A Comparison of PASCAL and FORTRAN As Introductory Programming Languages

Gary J. Nutt

CU-CS-101-77 January 1977



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

ABSTRACT

Many colleges and universitites offer introductory programming classes based on the FORTRAN language. Several of these schools are contemplating a change to a more modern programming language in this first course. The Department of Computer Science at the University of Colorado has recently made the transition from FORTRAN to PASCAL, and this paper offers an informal discussion of the experiences of one instructor during that change. It is hoped that others who may be considering a similar change will benefit from our experiences.

INTRODUCTION

A popular conjecture among computer science educators is that PASCAL or PL/C should be the first programming language taught in introductory classes. This theory is implemented at a number of schools, but the majority of introductory programming courses use FORTRAN. Recently the Department of Computer Science at the University of Colorado reorganized its introductory courses to eliminate FORTRAN in the first course in favor of PASCAL. In this paper, we discuss our experiences with this transition so that others may benefit from it.

Proponents of PASCAL (or PL/C) argue that it is a more modern programming language containing better control structures to encourage good programming techniques. The student is likely to develop a better appreciation for algorithm design and can more easily argue for the correctness of such algorithms if he tends to think of them in terms of PASCAL implementations. PASCAL is also alleged to be as easy to teach to the neophyte as FORTRAN, yet still be more elegant. PASCAL provides more general data types and an extensible data structure feature that clarifies the computations on data. Finally, FORTRAN can be easily learned once the student is comfortable with PASCAL although the converse is not true.

The FORTRAN camp offers the following arguments in favor of their language. FORTRAN is easier to teach (and learn) than PASCAL or PL/C, (a direct contradiction to the PASCAL camp). The number of implementations of FORTRAN far exceeds the number of implementations of any other language (with the possible exception of COBOL), thus FORTRAN is a more valuable tool to the novice programmer. PASCAL is not tried and tested by the real world of applications; it exists primarily in an academic atmosphere. There is a shortage of good textbooks for teaching PASCAL, while there is a plethora of FORTRAN-oriented texts available. FORTRAN can be written with style (as indicated by Kernighan and Plauger [6]).

In the following sections of this paper, the educational environment at this University is described, a subjective comparison of similar FORTRAN-based and PASCAL-based introductory courses is given with respect to the pros and cons mentioned above, and finally, our opinion of the two languages as introductory programming languages is given.

THE ENVIRONMENT

The Department of Computer Science is associated with the College of Arts and Sciences at the University of Colorado and offers the M.S. and Ph.D. degrees. A number of undergraduate classes are offered to support other programs such as the engineering curriculum and the undergraduate math curriculum. Prior to the Fall semester of 1976, three undergraduate introductory courses were offered to support various programs: C.S. 201 was an introduction to FORTRAN programming for scientists and engineers; C.S. 202 used COBOL to introduce programming to business majors; C.S. 203 was an introductory FORTRAN course for liberal arts majors. Each class used examples and assignments intended to appeal to students from the given discipline. During the 1975-76 academic year, the size of each section of each class was limited to 68 students with the idea of increasing the quality of lectures. Three sections of C.S. 201, one section each of C.S. 202 and C.S. 203 were taught. In Fall of 1976 all three courses were replaced by a PASCAL-based class entitled Fundamentals of Computing I. Students are expected to take a one year sequence starting with C.S. 210, followed by an additional semester dealing with larger programs and also introducing FORTRAN and COBOL. The philosophy for this reorganization was that programming is a sufficiently difficult intellectual activity that a full year is required to provide an adequate introduction. Furthermore, each student who learns programming, regardless of their background, can best appreciate the task by first learning a modern language like PASCAL before specializing in a particular application area. C.S. 210 was again limited to 68 students per section and five sections were offered during the Fall semester.

The basic topics covered in C.S. 210 are listed in Table I, and a summary of homework assignments used in the five sections is provided in the Appendix. The textbooks used for C.S. 201 were the Forsythe, Keenan, Organick and Stenberg book [3] and a FORTRAN text by Dimitry and Mott [2]. Conway, Gries, and Zimmerman wrote the text used in C.S. 210, [1].

SOME OBSERVATIONS

Is PASCAL more difficult to teach (and to learn) than FORTAN? It is not easy to answer this question conclusively based on the experiences, but it would appear that FORTRAN is slightly easier to learn than PASCAL primarily because there is less to learn. PASCAL introduces a number of programming concepts that are not possible in FORTRAN. For example, recursion was taught in C.S. 210 but not in C.S. 201; extensible data structures are introduced in PASCAL courses in order to discuss string manipulations, whereas FORTRAN strings are handled with integer arrays. (In the PASCAL case, the student learns to correctly type variables while in FORTRAN he is taught to bypass the typing mechanism.) PASCAL records are introduced in C.S. 210 but they have no counterpart in FORTRAN. A discussion of parameter passing mechanisms is necessary for PASCAL, since value and name calls are both permissible*. Additional constructs for controlling program flow are introduced in PASCAL. It was somewhat surprising to observe that the additional control constructs were not difficult for students to grasp, and most of them were as comfortable with the WHILE-DO, REPEAT-UNTIL, and FØR-DØ statements as the FORTRAN students were with the DO-statement. In certain cases, there is less material to learn about PASCAL then about FORTRAN. For example, the input/output operations have a decided edge in PASCAL since the need for FORMAT statements does not exist. Another area where less material is required in PASCAL is that of forming syntactically legal array subscripts and loop expressions; the generality of PASCAL is more easily learned.

Although the excess material contributes to the difficulty, the amount of extra material does not appear to be so significant that it makes PASCAL substantially more difficult to learn. Each course started with 68 students, and 44 students finished C.S. 201 (FORTRAN) while 46 students finished C.S. 210. 22 C.S. 210 students earned a grade of A or B, while only 16 C.S. 201 students earned a grade of B or better. This difference partially reflects the homework grades assigned by different Teaching Assistants in the two courses. The best measure of the comparative difficulties of the two languages is the amount of money each student spent over the semester on programming assignments. For the Control Data FORTRAN

* A corresponding difficulty arises in FORTRAN when the issue of passing literals via parameters that are redefined within the subprogram is discussed.

and PASCAL compilers, the cost of compiling and executing a simple job is about the same; C.S. 201 students used an average of $37.94 for the semester and C.S. 210 students spent an average of $32.41 for the semester.

Does PASCAL encourage better programming techniques through the use of better control structures? The intuitive observation here is that the C.S. 210 students were writing much better programs after a semester than the C.S. 201 students. Most of the PASCAL programs included loops that terminated normally witout resorting to an escape. When a student wrote a working PASCAL program, it was much easier to read than the FORTRAN programs from C.S. 201. Although the average amount of money spent on programming assignments was about the same for the two classes, it seemed as though the C.S. 210 students were spending less time programming than the C.S. 201 students. Most PASCAL debugging runs were made to correct missing semicolons and to balance BEGIN-END statements. It was also apparent that the programs produced by the C.S. 210 students were easier to show correct (or incorrect) than those written in FORTRAN.

If both languages are to be learned, what is the order in which they should be learned? It seems clear to us that it is much easier to learn FORTRAN if one is already familiar with PASCAL. The last week of C.S. 210 was spent on an introduction to FORTRAN. Although a few of these students had learned some FORTRAN or BASIC in high school, the majority were unfamiliar with the material. After the one week introduction, over 95% of the C.S. 210 students were able to write a FORTRAN program to compute the root of a predefined function using the bisection method. (The same assignment had been given in PASCAL earlier in the semester.) Previous experience with a second course in computer programming showed that C.S. 201 students needed to spend two or three weeks on an introduction to ALGOL before they could even begin to write equivalent programs.

CONCLUSIONS

From a computer scientist's point of view, replacing FORTRAN with PASCAL is a pedagogically sound improvement in the introductory programming class. However, the introductory class at the University of Colorado is a support class for other disciplines and is not part of an undergraduate computer science curriculum. This has led to some discontent among the faculty of the College of Engineering for the following reasons: The undergraduate engineering curriculum has so many required courses that a student may only be able to take one semester of programming. In that one semester, the student will only learn PASCAL. Most of the arguments in favor of FORTRAN apply to the engineering student, i.e., if only one language is learned it should be FORTRAN since it is implemented on most computers and it finds heavy use in the non-academic world. Some of the engineering faculty believe that only the most popular tool should be used even if it may be outdated by a more modern, but less available tool. Our experience with C.S. 210 seems to negate this argument to a large degree. The PASCAL students easily picked up a good deal of FORTRAN in only one week, and could apparently be comfortable with the language with little extra study. However, to become proficient in PASCAL or FORTRAN, the student will have to spend more than one semester writing algorithms and implementing them in a programming language. Programming is a difficult enough intellectual pursuit that it cannot be learned in a one semester class.

Kernighan and Plauger have indeed shown that it is possible to write FORTRAN programs "with style", [6]. Nevertheless, they too apparently believe that FORTRAN should be updated to improve the programs since they use RATFOR in their subsequent book on software tools, [7].

There are problems with using PASCAL in an introductory class, and these problems reflect both the environment in which PASCAL was developed and the relative youth of the implementation. The first problem is that the number of PASCAL oriented text books is severely limited. We know of only three books that could have been used in the Fall of 1976, (and one of them was not published until late in the summer of 1976), [1,5,8]. It is, of course, possible to use a

"language independent" book for most of the course, (e.g. references [3,4]), but a PASCAL manual of some type is still necessary. Hopefully, this situation will improve in the immediate future. A second problem has to do with PASCAL input operations. As long as only numbers are being read into the program, the READ (READLN) statements are adequate. Whenever character data is to be read, the students encountered nontrivial problems, primarily due the Control Data implementation. Once a character string has been read into a packed array of characters, it is cumbersome to determine the length of a left-justified substring stored in the variable. It would also be nice if the PASCAL implementation performed more run time type checking. The Control Data 6400 system has been primarily used for FORTRAN programming, thus most keypunches are 026 models. This caused PASCAL students to have to multi punch many characters such as colon, semicolon, brackets, etc.

In spite of some shortcomings, PASCAL provides an excellent medium for discussing several important features of programming languages and algorithm development techniques. Sorted binary trees were discussed in both C.S. 201 and C.S. 210; the FORTRAN coding of insertion and traversal algorithms required the student to use arrays to implement nodes and a stack. In PASCAL, the students were able to define node types and binary tree types (as records) and then to write recursive insertion and traversal procedures without worrying about a stack. Many C.S. 210 students were comfortable enough with recursive procedures that they were using them to produce Fibonacci sequences, etc. without being forced to do so. As one who learned FORTRAN as a first high level language, we were especially gratified at the ease with which recursion was learned in the introductory class.

Our overall opinion, after teaching C.S. 201 several times and C.S. 210 once, is that the C.S. 210 students did learn substantially better programming technique than the C.S. 201 students. Their ability to develop algorithms was improved somewhat, but the coding techniques showed the most significant improvement. We believe that the transition from FORTRAN to PASCAL has resulted in an improved introductory programming class.

---

* For example, PASCAL should not allow arbitrary integer values to be read into variables that have been typed with an integer subrange.

REFERENCES

[1] Conway, R., Gries, D., and Zimmerman, E. C., A Primer on PASCAL, Winthrop Publishers, Inc., Cambridge, Massachusetts, 1976.

[2] Dimitry, D., and Mott, T., Jr., Introduction to FORTRAN IV Programming, Holt, Rinehart, and Winston, Inc., New York, 1966.

[3] Forsythe, A. I., Keenan, T.A., Organick, E. I., and Stenberg, W., Computer Science: A First Course, John Wiley and Sons, Inc. New York, 1975, Second Edition.

[4] Gear, C. W., Introduction to Computer Science, Science Research Associates, Inc., Chicago, 1973.

[5] Jensen, K. and Wirth, N., PASCAL User Manual and Report, Springer-Verlag, Berlin, 1974.

[6] Kernighan, B. W., and Plauger, P. J., The Elements of Programming Style, McGraw-Hill Book Company, New York, 1974.

[7] Kernighan, B. W., and Plauger, P. J., Software Tools, Addison-Wesley Publishing Company, Reading, Massachusetts, 1976.

[8] Wirth, N., Systematic Programming: An Introduction, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

C.S. 210: Fundamentals of Computing I
Course Outline

Introduction
- The concept of an algorithm
- Refinement
- Flowcharting

Computer realization of algorithms
- Representation of data
- Computer hardware characteristics
- Assembly language

Procedure-oriented language
- Computations
- Control structures
- Structured data
- Procedures

Algorithms for non-numeric problems
- Sorting
- Searching

Algorithms for numeric problems
- Finding a root
- Finding an extreme

Table 1

APPENDIX: C.S. 210 Homework Summary

The semester extends over approximately 14 weeks. Of the five sections of C.S. 210, the number of homework assignments varied from 8 to 13. The following is a combination of typical assignments given in the various sections.

1. Obtain a listing of a permanent file containing lecture notes supplementary to the text book.

2. Write and test a simple PASCAL program. These programs varied from finding the volume of a house through finding the third side of a triangle using the Law of Cosines.

3. Write and test a PASCAL program that uses one or more of the control structures (i.e. IF-THEN-ELSE, CASE statement, WHILE-DO, REPEAT-UNTIL, FØR-DO).

4. Another assignment to apply control structures, e.g. sine approximation.

5. An exercise to introduce structured data (arrays), e.g. matrix multiplication.

6. A non-numeric processing assignment, e.g. print one month of a calendar given the number of the month, the day of the week of the first day in the month, and a leap year designation; recognizing palindromes; message decoding using a shift code; pattern matching.

7. Using procedures, e.g. rewrite the matrix multiplication program using procedures for input, output, multiplication; rewrite pattern matching program with procedures to determine string length, input a string; compute the root of a function, where the function is "externally" defined.

8. Write larger programs using procedures and functions, e.g. compute the biorhythm for a person given his birthdate; find the shortest route between any two of a set of cities; perform a bubble sort on an array of data.

9. Recursive programs, e.g. build and print a sorted binary tree; evaluate a given recursive function at multiple points.

10. A variety of assignments for various purposes, e.g. least squares fitting; FORTRAN coding; student grading program.