ED 048 907 LI 002 716

AUTHOR	Schipma, Peter B.; And Others
TITLE	Use of PL/1 in a Bibliographic Information Retrieval System.
INSTITUTION	Illinois Inst. of Tech., Chicago. Research Inst.
PUB DATE	15 Jan 71
NOTE	17p.
EDRS PRICE	EDRS Price MF-$0.65 HC-$3.29
DESCRIPTORS	*Computer Programs, Data Bases, *Information Retrieval, Information Storage, *Information Systems, *Programing Languages
IDENTIFIERS	SDI, *Selective Dissemination of Information

ABSTRACT

The Information Sciences section of IIT Research Institute (IITRI) has developed a Computer Search Center and is currently conducting a research project to explore computer searching of a variety of machine-readable data bases. The Center provides Selective Dissemination of Information services to academic, industrial and research organizations from chemical and biological data bases, using computer programs developed during the first year of the project. This paper discusses the preparation of those programs as it relates to the programming language, PL/1. The paper indicates that PL/1 is admirably suited to the needs of a system designed for search of bibliographic data bases. The most outstanding features of PL/1 in this regard are its flexibility, I/O capability and ease of programming, modification and documentation. The use of PL/1 has given an opportunity to do more true system research and development in less time than machine or assembly code would have, and also has been better for this type of work than other compiler languages. The use of PL/1 allowed development of a better overall system in terms of features and efficiency than could have been obtained by any other methodology. (Author)

Use of PL/1 In A Bibliographic Information Retrieval System

by

Peter B. Schipma, Research Scientist
Barbara Lynn Louthan, Assistant Mathematician
Barbara A. Boone, Technical Assistant

IIT Research Institute
10 West 35 Street
Chicago, Illinois 60616

15 January 1971

Use of PL/1 in a Bibliographic Information Retrieval System

by Peter Schipma, Research Scientist, Barbara Louthan, Assistant Mathematician and Barbara Boone, Technical Assistant
Information Sciences, IIT Research Institute

The Information Sciences section of IIT Research Institute (IITRI) has developed a Computer Search Center and is currently conducting a research project to explore computer searching of a variety of machine-readable data bases. The Center provides Selective Dissemination of Information services to academic, industrial and research organizations from chemical and biological data bases, using computer programs developed during the first year of the project. We would like to discuss the preparation of those programs as it relates to the programming language, PL/1.

The overall project objective was the development of a set of generalized computer programs for the search of text data bases. Since it was realized that there are many data bases which vary greatly in content and format, the programs had to be flexible enough to accept this diverse input. This flexibility required the use of modular programming techniques to allow a facility for rapid and simple program modification as new data bases were added.

Another of our goals was to have the programs available for use by others such as academic institutions and industrial organizations that had a sufficient number of in-house users to warrant efficient operation. Inasmuch as very few companies have this number of users, we have set up a central search operation at IITRI, called the Computer Search Center, which will continue to provide services to smaller companies, and, more importantly, serve as an operating organism which we can use as a test vehicle for research in information science. The programs were thus not to be designed strictly for our Center, but rather as a package that can be used by others.

A final goal was the design of a set of programs to meet the needs and desires of the potential users. We felt it necessary to include as many of the features which users found desirable as was practical; and, further, to make the programs easy to use and have their operation and limitations obvious to the users. The programs were written in PL/1 and are used to maintain an SDI service to about 200 users. We are also working toward the installation of the system in other organizations. We are continuing our research toward expansion of data bases, introduction of test and analytical data as well as textual information, and programming for a retrospective search system. We will not touch on most of these topics in this paper, since our emphasis here is to share some of our reasons for choosing PL/1 and some of our experiences with that language in this application.

Most of our project goals directed us toward the use of a high-level compiler language. We wished to be machine-independent, supplier-independent, and have flexible, understandable programs. In addition to these somewhat theoretical considerations, we were faced with more practical constraints, such as a programming staff comprised primarily of FORTRAN programmers, the availability of IBM 360 series hardware, and the potential use of a time-shared interactive system using a subset of PL/1 as a language. Also, and perhaps most importantly, we had a limited time in which to develop a set of programs that were open to a high degree of change, so that various techniques could be tested during the development of the system.

Before beginning the programming of the system, we surveyed the various similar systems then in use. From this survey we were able to draw the following conclusions.

- o Systems written in assembly language rapidly become rigid and hard to change in even minor respects. They are programmer-dependent and usually understandable only to the few people familiar with the author's particular programming techniques. Such systems are also extremely installation dependent.
- o Systems written in higher level compiler language were rare, fairly inefficient with respect to assembly language programs and generally suffered from lack of sufficient constructs in the language used (primarily FORTRAN and COBOL).

We found that a number of people were beginning to use PL/1 as a language more amenable to solution of the problems of text processing, and after a thorough investigation of that language, decided that it was the proper programming vehicle for our system.

The system now in operation at the Computer Search Center at IITRI definitely bears out some of the generalizations that have been made about PL/1.[1] It is less efficient than a machine or assembly language-<u>but</u> it allows such rapidity of programming as to afford the programmer an opportunity to try alternative strategies and perhaps make some basic discoveries in techniques that would not otherwise have been possible. It does mask the actual operations of the machine, but people desiring to use a system are often not interested in hardware and circuit design. PL/1 has additional advantages in terms of documentation and maintenance. We would like to discuss each of those points in terms of our experience.

[1]Mentele, James W. "Experience with PL/1 as an Information Retrieval Language" presented at the Annual Meeting of the American Chemical Society, September, 1970.

EFFICIENCY versus PROGRAMMING SPEED

Many people have compared the relative efficiencies of assembly language and PL/1, giving ratios of from 1:2 to 1:10, depending upon application. Indeed, some PL/1 statements, innocuous looking though they may be, give rise to extremely complicated and horribly inefficient object code. However, PL/1 is extremely rich in construct, and there are usually many ways of accomplishing a task, one or more of which is generally of fair efficiency.

Consider the somewhat simple case of character matching-coding an algorithm to detect the presence or absence of one character string within another. This type of algorithm is, of course, essential to a text search system - "does this word appear in that title?" The PL/1 built-in function INDEX can be used to create the required algorithm in one statement-but the object code generated is lengthy and very inefficient. The coding of the basic algorithm in assembly language is not too complex, and much more efficient. However, we wished to explore several characteristics of the English language that might be used to our advantage. Programming of a number of alternative strategies in a fairly short amount of time is possible with PL/1. In a matter of days, we had compared matching techniques based on one, two and three letters of each word being compared against the string, and found that for a term list of 2,000 to 20,000 words, matching based upon bigrams (letter pairs) was most efficient. We could not have done this multiple programming in nearly so short a time had we been using assembly language.

After the algorithm using bigram match had been in use for some time, some of our linguistic analyses indicated that use of the least common bigram in a word (rather than the initial bigram) would improve the efficiency of the algorithm.[2] For example, the word "MOLYBDENUM" would be searched on the basis of the bigram "BD" since that bigram appears less frequently in the data base than any other bigram in the word. Modification of the programs to test this theory was accomplished in two days. Without use of the compiler language we would have been unable to perform such a rapid check of a new idea.

When we programmed the various search strategies, we also varied the PL/1 instructions by which the matching was performed. We overcame the inefficiency of the built-in function, INDEX, by use of a very powerful PL/1 feature, overlay definition. Overlay definition permits multiple variable references to the same contents of core. For example, one may reference a set of 100 characters by a variable defined as a character string of length 100, or by a variable defined as an array of 100 elements each of one character, or by a variable defined as an array of 50 2-character elements, etc. We used overlay definition to enable us to treat a citation string as a whole, or to refer directly to any bigram in the string.

---

[2]A paper describing this analysis is being prepared by E. Onderisin of IITRI.

After this series of program modifications we still had less efficient object code that was generated by PL/1 - but had had the opportunity and time to develop a search algorithm that was an order of magnitude better than any use in text retrieval till then. The net result was a improvement in overall timing - the efficiency of the technique overcomes the reduced efficiency of the PL/1-generated code. Based on reported timings, our PL/1 programs are comparable in execution time to assembly languages programs designed for the same purpose that are now in use.

We think that this example illustrates one of the greatest advantages of PL/1. The programmer is free to try things without penalties in terms of man-days of time. Indeed, since he is removed from a good view of what goes on in the machine, he had better do some experimentation, since two slightly variant codings may have widely variant efficiencies. Surely a danger lurks therein, but if it is recognized it can definitely be used to advantage. Freed somewhat of the imposition of rigid coding constraints, the programmer can give more attention to the problem.

Also to be considered is the fact that the object code generated by PL/1 will probably not get less efficient than it is now. It has been gradually improving with successive versions of the IBM compiler (we have not as yet tried the new Optimizing PL/1 Compiler, which is supposed to be a phenomenal improvement) and should continue to do so. As compilers for computers other than IBM machinery are developed, overall efficiency of PL/1 should increase.

We should also note that the control given by PL/1 over interrupts is an extremely useful feature. In program debugging, it is a good tool; but more importantly in a production situation for which input is received from elsewhere (as in our case), the option of executing various portions of code depending on the interrupts which may or may not arise as the data base is read, is extremely important. Since we do not have control over the production of the data bases we use, it is very important to have the ability to program for various qualities of the data base which may give rise to interrupts. Branching based upon detection of an interrupt is a relatively inefficient operation, but one that does not have to be executed very often. It is a powerful tool for recovery within a production situation.

LOSS OF "POWER OVER THE MACHINE" versus MACHINE INDEPENDENCE

It is true that the use of PL/1 removes the programmer from the intricacies of the machine operation. In some cases this can be detrimental. For example, beginning PL/1 programmers sometimes pay very little attention to explicit declaration of variables, since the compiler will convert decimal to binary, numeric character to decimal, etc. However, all conversions take time, and some of them are very expensive, being done via a subroutine call and execution. Obviously, such expense should be avoided. In many cases, all that is necessary is more careful attention to variables, which requires only a generalized concept of the computer, and most programmers, even PL/1 programmers, do have this generalized concept. We feel that this generalized concept is sufficient. It will keep the programmer from making gross errors and yet he will not have to concentrate on hardware to any great extent.

The positive side of not needing to know much about the hardware is evident in program designs that approach machine independence. Here PL/1 shines. Although currently implemented only on the IBM 360 and 370 series of computers, PL/1 compilers are being written for Univac, CDC and Digital Equipment Corporation computers. It thus appears that PL/1 will someday be implemented on a wide variety of machines.

Even considering the IBM 360 family as a limitation in total, individually there is quite a bit of difference between a 360/40 and a 360/75. Yet we have run the same programs on these models and all those in between, with no problems, using both source and object decks, and a variety of peripheral devices.

One of the strongest features of PL/1 in this machine-independency aspect, is its highly flexible I/O capability. One can define files without further regard to the device upon which the files will be maintained other than whether the devices permit direct access or not. Both stream- and record-oriented files are permitted and most of the file description can be relegated to the operating system if desired.

Of particular interest to the text processor who must handle large amounts of data are the following features of PL/1.

- <u>Structure variables</u>. Variables in PL/1 may be related in hierarchical arrangements. The elements of such a structure may be of differing data types (binary, character, bit, etc.) and may be arrays or other structures (or even arrays of structures or structures of arrays!). A structure may be manipulated as a whole, or by any of the elements it contains. One example of the use of structures would be for the elements of a bibliographic citation (author, title, pagination, etc). The entire structure can be read into core and then each element can be manipulated separately.

o <u>Dynamic storage allocation.</u> An area of core storage can be used repeatedly for different data by successive allocation and freeing of the area with varying characteristics being defined at each allocation. Also, core storage can be requested during execution based on parameters generated by the program. In one of our programs an amount of core storage is allocated based upon evaluation of a quadratic equation. The variables in the equation change for each set of data, and so the equation is reevaluated and a different amount of storage allocated for each set of data.

o <u>String handling functions.</u> There are several functions in PL/1 that are expressly designed for manipulation of bit- and character-string data. There are no such direct functions in FORTRAN or COBOL, although most of the operations can be achieved in those languages. These functions provide immediate, one-instruction capability, to do such things as find the current length of a string, concatenate strings, determine the position of a given configuration in a string, extract a portion of a string, translate strings, and so on.

o <u>Based storage.</u> Data can be maintained in input buffers rather than read into core, and effectively manipulated by treating pointers to the data rather than the data itself. This permits savings in both storage space and time. The presence of pointer variables in PL/1 also permits list processing.

All of these features, as well as many others, are powerful tools that can be used without a clear image of what is happening in the hardware. Again, we feel that they lend themselves to programming freedom rather than constraint.

<u>Documentation and Modification</u>

The statement "It is finished!" does not belong in a programmer's, or system designer's, vocabulary. It cannot, for things change too rapidly these days both in terms of hardware capability and system requirements. Therefore documentation and ease of modification have become increasingly important. It is in these aspects that higher-level languages far outstrip the admittedly more efficient machine and assembly codes, and the economics of program updating may more than compensate for the loss in efficiency.

PL/1 is among the best of the higher-level languages in this respect since it is procedure-oriented. If a system is originally built in a modular fashion using PL/1 procedures, it is fairly easy to make modifications and incorporate them into the documentation. In our search system, for instance, there are separate programs for data base preparation, profile preparation, search, and output generation. Information is passed from each main group of programs to the next via files, and within groups via parameters and corresponding arguments. Thus a change within a group may affect two or three procedures, but will not affect another group unless a file needs some modification. Documentation can then be treated in building-block fashion, describing each program group separately and tying them together by means of the files by which they communicate. Also, even within procedures, programming may be modular, using the block structure of PL/1. This allows rapid modification of sections of procedures without repercussions from unexpected places.

Summary

In total, we have found PL/1 to be admirably suited to the needs of a system designed for search of bibliographic data bases. The most outstanding features of PL/1 in this regard are its flexibility, I/O capability and ease of programming, modification and documentation. We feel that use of PL/1 has given us an opportunity to do more true system research and development in less time than machine or assembly code would have, and also has been better for our particular type of work than other compiler languages. The use of PL/1 has given us a better overall system in terms of features and efficiency than we could have obtained by any other methodology.

ABSTRACT

The Information Sciences section of IIT Research Institute (IITRI) has developed a Computer Search Center and is currently conducting a research project to explore computer searching of a variety of machine-readable data bases. The Center provides Selective Dissemination of Information services to academic, industrial and research organizations from chemical and biological data bases, using computer programs developed during the first year of the project. This paper discusses the preparation of those programs as it relates to the programming language, PL/1.

The paper indicates that PL/1 is admirably suited to the needs of a system designed for search of bibliographic data bases. The most outstanding features of PL/1 in this regard are its flexibility, I/O capability and ease of programming, modification and documentation. The use of PL/1 has given an opportunity to do more true system research and development in less time than machine or assembly code would have, and also has been better for this type of work than other compiler languages. The use of PL/1 allowed development of a better overall system in terms of features and efficiency than could have been obtained by any other methodology.

Keywords and Phrases

PL/I

Compilers

Programming Languages

Information Storage and Retrieval

Text Searching

Higher-level Languages

Bibliography

Corbato, F.J. "PL/1 as a Tool for System Programming: Datamaticn, May, 1969

Paquette, Russell "Software" The Art, the Science, the Industry SDC Magazine Vol. 11, No. 3; March, 1968

Rubey, Raymond J. et al "Comparative Evaluation of PL/1" Report CS-6813-R0106, ESD-TR-68-150, Logicon, Inc. April 1968

Schwartz, E.S., Williams, M.E. (IIT Research Institute), and Fanta, P. (Illinois Institute of Technology). "Modern Techniques in Chemical Information (Workbook and Syllabus)," February, 1969. To be published.

Shaw, Christopher J. "The Utility of PL/1 for Command and Control Programming "Presented at PL/1 Seminar sponsored by Logicon, Inc., December 5, 1967 at USAF Electronic Systems Division, Hanscom Air Force Base, Bedford, Massachusetts

Walter, Arline and Bohl, Marilyn "PL/1 Programming Aids" Software Age, November 1969

Williams, M.E. and Schipma, P.B. "Design and Operation of a Computer Search Center for Chemical Information" published in the Journal of Chemical Documentation, Vol. 10, #3, p. 158, 1970.