TWENTY FIVE YEARS OF FORTRAN

A National ACM Lectureship
Series Presentation
by

J.A.N. Lee
Pioneer Day Chairman
National Computer Conference 1982

Department of Computer Sceince
Virginia Tech
Blacksburg VA 24061

82/11/12

CS82010

Abstract

In 1982 FORTRAN will have existed in the environment of computers, computing and computation for 25 years, making it one of the most successful of programming languages even if it is not the actual oldest still surviving language. The honor of being the oldest still belongs to APT (Automatic Programmed Tool.) This report is the script of talk given at several institutions during the Spring of 1982 and serves as a skeleton on which a broader history is to be developed.

IBM was a comparative latecomer in the electronic computer market, T.J. Watson Sr. having little confidence in the reliability of machines which one could not actually see working, and perhaps influenced in some degree by the prognosis that a small number of such machines would satisfy the world's demands for computation. While it is true that IBM had supported Howard Aiken in the development of the (Harvard) MARK I, that too was a relay machine because IBM was marketing mechanical devices; if Munroe [1] had been the sponsor then perhaps the MARK I would have been electronic. Of course under those circumstances IBM would not have been injected into the electronic computing field any earlier. At the same time, when IBM did eventually enter this field with the SSEC, the emphasis was on providing hardware and supplying the customer with programming support, at a fee, through the service bureaus. Thus to provide "software" (as we know it today) would have counter to the profit motives of those bureaus and the day had not yet dawned when programs were for sale! User cooperatives were just beginning to emerge as a means by which customers could freely exchange their in-house programs.

Speedcoding - the floating point simulator

After the advent of the IBM 701, some of the drudgery of keeping track of the radix point in fixed numeric fields was relieved by the introduction by John Backus of the Speedcoding system. This

---

[1] See foreword by Bernard Cohen to: "History of Mechanical Computing Machinery", by George C. Chase, Ann. Hist. Comp., Vol.2, No.3, July 1980, p.198.

system made the 704 appear to be a three-address floating point calculator with the added advantages of input-output conveniences. While it was an interpretive system, the reduction in costs of coding, testing and operation proved it to be a more economical means of problem solving than direct machine language programming [2].

The IBM 704 -- built-in floating point and indexing

The design of the next machine for the IBM family of electronic calculators originally did not include any provision for hardware floating point arithmetic or indexing operations. Backus championed this cause against strong opposition which was based on the profit motives of not giving away too much of the computational pie; after all, such niceties would only decrease the time needed to complete a computation in the service bureau thus decreasing the profit to be gained. The battle was won on the side of improvements in hardware services and as a result such systems as Speedcoding were no longer viable. Backus [3] stated that "... early systems ... had hidden a lot of gross inefficiencies ... in floating point routines ... (and) clumsy treatment of looping and indexing ...", so that when the 704 came along with its hardware floating point and indexing "... there was just nowhere to hide inefficiencies."

WHY FORTRAN?

---

[2] Backus, J.W., The IBM Speedcoding System, Jour. ACM, Vol.1, No.1, January 1954, pp.4-6.
[3] Backus, J.W., FORTRAN Session, History of Programming Languages, Academic Press, New York, 1981, p.50.

Thus IBM needed to look towards means of producing programs automatically which were at least as efficient as those that would be written by hand and preferably using a language similar to that that had been proposed by Rutishauser [4] and hinted at by such authors as Glennie [5].

Automatic Programming

At this time (1953) automatic programming was regarded as the wave of the future though in general the attempts at design and implementation were somewhat narrow in their scope or foresight. The concept of machine independence was lacking and the point of using a language which was non-machine-like seemed to have been missed. There were many sceptics who believed either that the task was impossible (see [6]) or that it was beyond the current state of the art.

THE PAPER LANGUAGE

December 1953 Memorandum Backus to Hurd

To propose the development of an automatic programming system which was not only more advanced in its language concepts as well as its ability to produce efficient code was the daring step which Backus proposed to his manager, Cuthbert Hurd, in December 1953. Without either a pre-authorized budget and a supporting staff or a detailed proposal, Backus was given the go-ahead for

---

[4] and [5] see Knuth, D.E. and Pardo, L.T., Early developments in programming languages, in Encyclopedia of Computer Science and Technology, Dekker, New York, Vol.7, pp.419-493.
[6] Hopper, G.M., The Early Days, in The History of Programming Languages, Academic Press, New York, 1981, p.13.

PRELIMINARY FORTRAN (Nov. 1954)

The preliminary proposal prepared in May 1954 does not actually mention the name FORTRAN, that being left for a Preliminary Report in November 1954. The significant elements of that report include such items as:

2 character variable names
function names 3 or more characters
multiply -- x , involution -- xx
relative constants

where relative constants where attributes which could be ascribed to identifiers whose associated values were to be "relatively constant".

Statements

Two statements in this preliminary report are significant for both their ingenuity and foresightedness, even though neither appeared in the resulting compiler to be delivered in 1957:

DO 10,14,50 I = 4,20,2

The significance of this statement is the three statement identifiers in the prefix to the loop control information; the meaning is that the block bounded by the statements labelled 10 and 14 should be repeated until the loop control conditions are exceeded, following which the next statement to be executed is that labelled 50! Since there was no requirement that the DO statement be contiguous to the block of statements to be

this research project. Perhaps Hurd didn't really expect the project to last too long, since in those days it was unheard of to spend "man-years" on writing programs. Actually he expected the task to be completed within six months.

The Initial Proposal 1954

A mid-1954 initial proposal suggested that the programmer "... would like to write (the mathematical formula) instead of the ... instructions (for) this expression ...".

Backus, Ziller and Herrick

Who were these visionaries who were to succeed where no-one else had ventured? The 1957 biographies state: "John (Backus) ... joined IBM in 1950 as a programmer in the Pure Science Department working with the Selective Sequence Electronic Calculator (SSEC). He transferred to the Scientific Computing Service in 1952 (and) ... in 1954 he was appointed Manager of the Programming Research Group in Applied Science and is presently the Department Manager of the Programming Research Department. John was awarded BS and MS degrees in mathematics by Columbia University ... (his) hobbies are hi-fidelity and chess."

"Harlan (Herrick) ... was raised in Iowa ... (and) came to IBM eight years ago (1949) from Yale where he taught mathematics. He received his masters at the State University of Iowa ... is a member of Phi Beta Kappa, Sigma Xi and the Masthematics Association of America ... (his) pet peeves are materialism, dishonesty and hypocracy ..."

Regrettably no such biography exists for Irving Ziller though he joined IBM in February 1952 and was the first to join Backus on

repeated, then this might be termed a remote loop specification. Obviously this complexity was muted by the time the first implementation was completed but it is interesting that similar statements did exist later in JOSS and COBOL (PERFORM...VARYING), and is similar in its use of a remote block of code to the BASIC GOSUB statement. It is important to remember that subprograms were not "invented" [7] in this form until two years after this preliminary design, though Mauchly had mentioned the subroutine concept in 1948 [8].

IF (X>Y) 12,55

Most people relate the three-way arithmetic IF statement with the original FORTRAN, but surprisingly enough that statement was a replacement for the much more modern statement shown above which did not re-appear in the language until FORTRAN IV in 1961. Similarly, the use of mathematical symbols such as > and thus the implication of logical expressions was left to a later version of the language. Quite distinctly, these symbols did not exist on the standard IBM key punches of the 1950's era.

Relabel

The Relabel statement was the beginning of a concept which has not yet been reintroduced into FORTRAN, that is array processing.

---

[7] Wheeler, D.J, Wilkes, M.V., and Gill, S., The Preparation of Programs for an Electronic Digital Computer, Addison-Wesley, Reading MA, 1957.

[8] Mauchly, John W., Preparation of problems for EDVAC-type machines, Proc. Symp. on Large Scale Digital Calculating Machinery, 1947 January 7-10, reprinted in Randell, B., (Ed.), The Origins of Digital Computers, Springer-Verlag, New York NY, 1982 (Third Edition), pp.393-397.

The intention of this statement was to permit the programmer to relabel the rows of a matrix so as to rearrange the row ordering and thus apply a single algorithm to a particular slice of that array. The complexity of this operation was not realized at the time of the preliminary report and it is clear that indexing through arrays explicitly is a much better programming technique than an implicit set of possibly unreadable instructions.

Frequency

When considering that one of the major objectives of the research project was to prove that machine could generate code on a par with a human programmer, the inclusion of information on the expected frequency of execution of statements is a logical necessity. However, it was to be found that such statements were unnecessary since logical flow analysis could provide the same (if not better) information.

FUTURE FORTRAN (from a 1954 perspective)

The preliminary report was not bashful in suggesting that the language might one day be extended to include new facilities.

begin-end

While the bracketing of a block of code by the reserved words begin and end is usually associated historically with ALGOL, this preliminary report included these terms as scoping delimiters for different types of arithmetic to be performed.

complex, double and matrix

It was proposed that in subsequent versions of FORTRAN it would

be possible to prescribe the type of arithmetic to be performed in certain segments of the program. It is not stated whether these operations would be associated with new data types or whether the first character in a name would signify different types than previously used.

CHANGING ENVIRONMENTS

From the beginning of the development period, the Programming Research group were shuffled through various locations in the area of 590 Madison Avenue, there perhaps being a correlation between the quality of the facilities and the recognition given to this project by the IBM administration.

Administrative attitudes

Chess in the afternoon or punch the time-clock?

Hal Stern [9] remembers clearly a steadily changing atmosphere through the period, beginning with a very "researchy" environment when people worked as hard as any other time but where the actual time of day was irrelevent. To him it was not unusual to work hard for a period, to play a game of chess and then to return, refreshed, to the task at hand. Thus when, apparently under pressure from upper level management, Backus broke up a chess game between Stern and Peter Sheridan, the former was somewhat incensed; as he remembers it was the first time he was winning!

Changing locations and the outlook

Backus believed that the changing locations distinctly affected

---

[9] Personal Correspondence

the productivity of the group, and there seemed to be a correlation with the view from the windows [10]:

> "... We shifted around from one small building (to another) at fairly regular intervals and this seemed to affect our work habits in a strange way. ... The first one we moved to overlooked the dressing rooms of the J. Thorpe department store and then we moved to another overlooking the dressing rooms of Bonwit Teller. ... (This was) a period in which our productivity seemed to decline considerably. From there we moved to a building on 56th street ... I noticed that when I came in everybody was there and apparently had been there for some time. (Eventually) someone confided to me that across the street ... was a young lady ... who slept without any clothes on ... and (who) danced very exuberantly ... before going to work. This was a period of great productivity because everybody came in early and after the show was over settled down to work long before (the official) starting time."

THE VON NEUMANN CONSTANT

Cuthbert Hurd had originally expected that the FORTRAN project would have been completed within six months of its starting date. But he did not realize that the von Neumann constant applied to the project. This constant is defined as being the time to completion of a project from the instant the enquiry is made

---

[10] From the Transcript of the Anecdotes told by numerous programming language pioneers at the History of Programming Languages Conference, Los Angeles CA, 1978, unpublished.

regarding the completion of the project and is a constant (usually about six months).

Missed delivery dates and changing machine requirements

An examination of successive SHARE meeting minutes starting in 1955 reveals that FORTRAN was to be delivered by the time of the next SHARE meeting (six months away) and that to meet this objective a little more machine was needed each time. When finally delivered the minimal machine configuration was 4k (IBM 704) words of core memory, 1 drum unit and 4 magnetic tape drives.

VON NEUMANN and FORTRAN

During the early years of the 1950's, von Neumann was hired by Cuthbert Hurd as a 30-days per year consultant mainly to assist with the design of the 704 and later to consult on mathematical problems. Apparently the technique was to establish von Neumann in an office in the World Headquarters and then to have those with problems bring them to him for consideration. Typical of the quickness of mind of von Neumann is the report of one of these consultations which involved John Greenstadt [11].

> "I did some hand calculations ... and it converged in the few cases I tried ... (so) I tried to prove convergence ... but with no hint of success ... Finally in the latter part of 1953, we decided to ask von Neumann (for help) ... I explained it to him in two minutes ... He spent the next

---

[11] Goldstine, H.H., Footnote to a recent paper, Jour. ACM, Vol.7, No.1, January 1960.

> fifteen minutes thinking up all the approaches we had thought of in three or four months, plus a few new ones ... at this point he decided it was a non-trivial problem and perhaps not worth it anyway ... and immediately suggested ... the truly natural generalization of (the) method."

1954 Meeting -- Von Neumann, Backus, Hurd and Beckman

Thus it was natural for Hurd to suggest that von Neumann's opinion of FORTRAN might be worthwhile -- after all it would cost no more than a day's presentation! Besides Backus and Hurd, they were joined by Frank Beckman, then manager of "Pure Programming". Beckman reports the conversation in his book [12] and in personal correspondence:

> "I do not know if von Neumann expressed any opinion about FORTRAN outside of this meeting, but I would certainly not describe his reaction at the time as being unduly negative -- somewhat apathetic perhaps, but not strongly negative. I remember very vividly his allusion to Turing's "short code" ... In general von Neumann was not an enthusiastic of automated programming aids ... I have always felt that since he, himself, did not require such aids in writing programs, he could not empathize with the typical production programmer."

In effect, von Neumann's response to the presentation on FORTRAN was "Why would you want another language?"

---

[12] Beckman, F.S., Mathematical Foundations of Programming, Addison-Wesley, Reading MA, 1980, pp.177-178.

Perhaps it is ironic that some twenty years later Backus chose for the title of his Turing Lecture "Can Programming be Liberated from the von Neumann Style?" [13].

1957

Eventually, the year arrived when FORTRAN was once more due to be delivered, but now the working system refused to be duplicated. The problem was that it was the practice to distribute programs through SHARE as card decks but this was probably the largest program ever intended to be distributed up to that time, and the card punches refused to remain stable enough to complete the punching of a single deck. Thus it was decided to distribute the compiler on magnetic tape instead.

Actual delivery

No record exists as to when the first compiler was actually shipped intentionally; perhaps part of the problem was that the US Post Office closed down a part of its operations about the time that it should have been shipped and was waiting for more money from Congress before they adjourned for Easter!

The way it was that week

As we can show later, it would appear that the first deck was shipped during the week of April 15-20, 1957. That was a week during which many things were reported in the New York Times:

-- Britain and Egypt were at war over the Suez Canal

---

[13] Backus, J., Can Programming Be Liberated from the von Neumann Style?, CACM, Vol.21, No.8, August 1978, pp.613-641.

-- Nasser of Egypt instigated a coup against King Houssein of Jordan but failed

-- Ike was President and Dick was his VP

-- the Yankees beat the Dodgers 5-1 in the Sunday game -- the Brooklyn Dodgers that is

-- Studebaker-Packard announced that they intended to offer a new small car to sell at less than $2000 and for the first time it would offer as standard features items which had only been optional before -- a heater, a defroster and directional signals!

--The Chrysler Corporation announced first quarter earnings of $1,100,000,000

-- the Canadiens beat the Bruins for the Stanley Cup again

-- the Dodgers announced plans for a new stadium in Flushing Meadows

-- Dean Martin's guest on his first TV show was Bing Crosby

-- Desilu Productions were confident that "I Love Lucy" would be back next year

-- Poland was warned against Western aid by Krushchev

-- NBC was planning to put TV shows on tape instead of running them "live"

but the delivery of the first FORTRAN compiler and the running of the first program escaped unnoticed. Perhaps the week itself is not crowned with glory -- it was the same time in 1912 (45 years previously on April 15th) that the Titanic sank!

INNOVATIONS

It is difficult to pick out any single item which makes FORTRAN unique by itself; almost everything that was delivered in that first package was innovative either because it was the first time that the feature had been placed at the disposal of a user of a high level language, or because in conjunction with other features of the language or the compiler it was an outstanding contribution to the science of computation.

FORMAT

The concept of being able to specify the format of an input or output item was not new; in fact the FORTRAN implementer of FORMAT, Roy Nutt, had previously included a similar system in an internal system for United Aircraft of Hartford CT.

Optimization techniques

In some respects, the development of language was incidental to the research to prove that a machine could produce good every bit as good as that produced by a human programmer. Backus [2] states that "... the degree of optimization they achieved was not really equalled again in subsequent compilers until the mid-1960's when the work of Fran Allen and John Cocke began to be used ..." This optimization was so good in fact that the "proprietors" of the optimization section (Irv Ziller and Bob Nelson) often thought that the results were wrong the generated code being unrecognizable as having originated in the code they input! Regrettably, it was not the practice to document as one went along in this age of compiler development and thus it was not until some years later that any of their techniques were

published [14].

User's manuals

While not significiant to either the design of the language or the implementation of the system, the introduction of a readable user's manual and a programmer's primer was clearly significant to the ultimate success of the language. The fact that the original user's manual described the whole of FORTRAN, with examples in less than 50 pages while at the same time providing wide margins for the keeping of notes by the reader, is both remarkable and a feat which has not been repeated again. Interestingly enough, the von Neumann constant apparently also relates the publication of the Programmer's Reference Manual and the actual delivery of the compiler -- the manual is dated October 15, 1956 and the compiler was released on April 15, 1957.

THE FIRST PROGRAM

The first error message

The running of the first program, though well documented by Herb Bright [15], was not a planned activity. There is one error in Bright's report which needs correction -- 1957 April 20 was a Saturday not a Friday as reported. It was on the afternoon of that Friday when an unmarked deck of cards was delivered to Westinghouse-Bettis and which was assumed by Lew Ondis to be the

---

[14] Lowry, E., and Medlock, C.W., Object Code Optimization, CACM, Vol.12, No.1, January 1969, pp.13-22.
[15] Bright, H., FORTRAN comes to Westinghouse-Bettis, 1957, Computers and Automation, November 1971, pp.17-18.

right size to be a FORTRAN compiler. Jim Callaghan quickly wrote a small program based on a recent technical report by Ollie Swift, and using the "common" technique for running programs on the IBM 704, the unmarked deck was loaded into the syetm followed by the program. Surprisingly it worked and in a short time the first FORTRAN error message was output:

FORTRAN DIAGNOSTIC PROGRAM RESULTS

05065 SOURCE PROGRAM ERROR. THIS IS A TYPE-GO TO ( ),I
BUT THE RIGHT PARENTHESIS IS NOT FOLLOWED BY A COMMA

END OF DIAGNOSTIC PROGRAM RESULTS

The error was quickly fixed and the program (apparently) recompiled and executed to produce "... a whiff of computing followed by 28 pages of output ..."

IMPROVING THE CODE

One of the flavors of computing in those days was the belief that almost anyone could produce code better than IBM could. Backus [2] quotes Perlis as wondering "... why those clods working on FORTRAN had taken 25 man years to produce a compiler, since one of his graduate students had written an IT compiler in a single summer ...!" Perhaps Perlis did not understand the complexities of code optimization.

Frank Engel -- Westinghouse-Pittsburgh

Thus when Engel noticed that during the compilation process there

was never any instant when two tapes were in use on the 704, he asked his account representative (Ken Powell) for a copy of the source code. Powell relayed the request to his manager, Frank Beckman, who responded "IBM does not supply source code." Not to be outdone, Engel dumped the compiler code (in octal), spotted the section which was responsible for this tape management, rewrote it (in octal) and produced a system which had an improved throughput of about 2 to 3 times the speed. When Powell saw this he asked Engel for a copy which he could send back to the FORTRAN group in New York -- Engel responded "Westinghouse does not supply object code".

THE PEOPLE OF FORTRAN

A complete dossier on all the members of the FORTRAN team (John Backus, Harlan Herrick, Irving Ziller, Robert Nelson, Roy Nutt, Peter Sheridan, Lois B. Mitchell Haibt, Sheldon Best, Richard Goldberg, David Sayre and Grace (Libby) Mitchell) is not possible here. Their individual contributions to the implementation have been documented by Backus [2]. But what of the individuals?

Backus, Nutt, Haibt

Backus has maintained his leadership in the field of computing through the years and is somewhat frustrated that the success of FORTRAN both overshadows and perhaps even thwarts his efforts to improve programming languages [13]. He has been recognized by the IEEE in 1967 (on the tenth anniversary of FORTRAN) with the W. Wallace McDowell Award, by the United States of America in 1976 (the bi-centennial year) with the National Medal of Science,

and finally by ACM in 1977 (on the twentieth anniversary of FORTRAN) with the Turing Award. Having completed the FORTRAN implementation, Backus was appointed by John Carr, ACM President, to be a member of the joint ACM-GAMM group which developed ALGOL. A close examination of the ALGOL proposal and the FORTRAN preliminary report reveals that perhaps it was Backus who introduced certain salient language features, such as begin-end, which were not included in FORTRAN. Backus, with the editorship of Peter Naur, also invented the syntactic definition schema now universally known as BNF, and variously as Backus-Normal-Form or Backus-Naur-Form.

Roy Nutt was not a member of the IBM staff which was assembled to develop FORTRAN, but instead was a highly knowledgable user whose concepts on input-output, and especially FORMAT, could not be duplicated elsewhere. Thus with the support of Walter Ramshaw, his manager at United Aircraft in Connecticut, Nutt spent a few days each week in New York assisting the IBM team. Like Backus, Nutt has remained agile in the field of programming languages and was recently embroiled deeply in the Ada Programming Language controversy [16].

Lois Haibt, while a contributing member of the team, is probably notable for being one of the few computer scientists ever to be featured in the Madamoiselle Magazine [17]:

"This twenty-two-year-old girl started at IBM with a salary

---

[16] See ACM Forum, CACM, Vol.24, No.11, November 1981, p.784 and succeeding issues.
[17] Kirkbride, K., and Garland, K., Machine, What do you think?, Madamoiselle, October 1958, pp.92-157.

> of five-thousand dollars a year and increased her income to six thousand in eight months ... Lois spends a good part of her day at a large bare desk writing up instructions for the computer to follow ... (she) waits her turn at the machine in a glass enclosed, red-walled balcony above it ... The girl who keeps the computer's social calendar tells Lois to stand by, ready, so that when the person before her is finished Lois can step up for her "date" immediately."

THE LANGUAGE MANUAL AND THE PRIMER

As stated previously one of the primary innovations which accompanied the FORTRAN implementation was the user's manuals which the group provided. These manuals are significant both from their conciseness and clarity, but also from their form. Apart from the PRINT 1 manuals which were published the same year, and the FOR TRANSIT manual of 1957, they are unique in the cover design (the team voted democratically on it) and for the fact that all these manuals contain the names of the authors. Like FORTRAN itself, these manuals were so good that they inhibited other publications on the topic and it was not until four years later that any form of competitive publication was available.

FOR TRANSIT

If we can do it once, we can do it again

In 1982 it is common sense that once one has produced the first version of a product the second version has got to be better. In

fact, one of the proverbs of programming [18] states:

Don't Be Afraid to Start Over

So when IBM introduced the 650 computer in the midst of the fever of user's anticipations regarding FORTRAN, it was obvious that another implementation was needed.

Novice crew

The 704 team led by Backus was still very busy in early 1957 completing the final stages of debugging and trying to get the system punched ready for distribution. Thus no-one could be spared from that group to start a new project, but there were people who had been close enough to the activity who could parallel their work. One of those was Bob Bemer. He assembled a 650 team consisting of Otto Alexander and David Hemmes at the Langdon Hotel on 56th Street, neither of whom had any previous experience with the kind of work to be undertaken; later they were joined by Flo Pessin who was equally unprepared for the task.

Cascading Implementation

Bemer noted that there was another significant activity in progress which was to implemented on the 650, though it was not originally intended for that machine; it was the IT (Interpretive Translator) system being developed by Alan Perlis at Carnegie Institute. IT compiled a much simpler language into the assembly language of the IBM 650 (SOAP - Symbolic Optimized Assembly

---

[18] Ledgard, H.F., Programming Proverbs for FORTRAN Programmers, Hayden Book Company, Rochelle Park NJ, 1975

Program) which in turn was assembled into the object code for the machine. Thus the concept was developed to "cascade" the implementation from FORTRAN to IT to SOAP and hence to object code in four passes.

FORTRAN > IT > SOAP > OBJECT

Quick implementation

In spite of the inexperience of the crew, a version of FORTRAN was available for the 650 only a few months after the delivery of the 704 version and the expenditure of only 4-5 man-years of work. To accomplish this the language was a subset of that implemented for the 704, but this was consistent with the fact that the 650 was an even smaller machine than the 704. Part of the motivation for this effort was the fact that IBM expected many more 650's were expected to be installed in Universities than was the case for 704's and especially since IBM was now willing to offer a 60% educational discount to those institutions which used the systems for administration, scientific computation and business data processing classes. FORTRAN was to be the "hooker" of this new generation of students.

To accelerate the implementation, and partially in recognition of the success of the language for its own right rather than simply as a test bed for optimization research, optimization was omitted from this new implementation. A form of optimization existed in the SOAP system, but this was not language dependent and was merely concerned with the location of instructions on the drum

(the main store of the 650) in relation to the pertinent data and/or the next instruction.

Bemer and Pessin

Bemer joined IBM Programming Research in 1955 after a career in the aircraft industry and was appointed manager of Programming Systems in 1960. In the "FORTRAN years" Bemer was active in many other ways which furthered the development of programming languages. He created the first load-and-go compiler for PRINT 1, developed the language known as COMTRAN (Commercial Translator*) and actively supported the development of ALGOL. In fact, in 1960 he was quoted as stating to the British Computer Society that "... we wish to obsolete FORTRAN and scrap it, not perpetuate it. Its purpose has been served."

Flo Pessin was given the task, by Bemer, of writing the arithmetic scanning routines for this new version of FORTRAN, but first she invented the name of the system -- FOR TRANSIT. Based on the cascading approach that Bemer had suggested, recognizing the contribution of IT and being a double-crostic addict, she coined the name as a three-way pun. One of the difficulties facing Pessin at this time was both her lack of experience in preparation for this task, the fact that the 704 team had created no documentation (though they were no different than most other implementers) and there was no help offered by the 704 group. Thus she was forced to invent new techniques of compilation, and

---

* It is interesting that the name is reminiscent of the source of FORTRAN -- Formula Translator.

like the others omitted to document them because she really did not know that what she was doing was so innovative.

INNOVATIONS

The problem of how to analyze and then generate code from an arithmetic expression was solved in a highly ingeneous manner. Pessin recognized (probably from high school) that the order in which operations were to be executed in an arithmetic expression was determined by the hierarchial order of the operators. Further the order of execution of fully parenthesized expressions is determined by the depth of parenthesizing. Thus the technique developed was to introduce into the expressions additional parentheses surrounding the operators, but facing outwards away from the operators, such that the number of parentheses added was in inverse proportion to the hierarchy of the operator. Sufficient additional parentheses were made available at the ends of the expression to satisfy the parenthesis balancing requirements. Thus given the expression:

a + b * c

the first stage of parenthesizing would produce:

(...(a))+(((b))*((c)...)

which after cancellation of parentheses surrounding the operands develops the expression:

(a + (b * c))

which is correct. Obviously this technique does not take into

account left (or right) associativity, but that is unimportant once this stage has been reached. The next innovation which was introduced was the use of a tabular method of expression analysis which was later rediscovered by others [19].

FIRST FILM

The first film, known to us, was produced by the New York Education Center (of IBM?) and is interesting for several reasons.

Not only is it interesting from a content point of view but also visually and editorially. The commentator is architypical of the "IBM salesman" of the era -- grey suit, white shirt, dark tie and the neatest of haircuts. The presentation is similarly in a style which we associate with both the corporation and the era; looking straight out, unsmiling and sincere. The first sentence is perhaps a commentary on where it was thought the concept of programming languages was heading:

> "... FORTRAN represents the most advanced coding system available today and is a forerunner of a universal coding language toward which we are working ..."

John Backus would be very interested in that statement since he is still working towards that goal. A later statement regarding the effort and cost of the development process is the first which we have recorded:

---

[19] See for example: Samelson, K., and Bauer, F.L., Sequential Formula Translation, in Programming Systems and Languages, Rosen, S. (Ed), McGraw-Hill, New York, 1967, pp.206-220, originally published in German in 1959.

"... was developed ... at a cost of $475,000 and ... 29 man years ..."

Since the film refers both to FORTRAN for the 704 and FOR TRANSIT for the 650, perhaps this breaks down to 25 man-years for the Backus project and 4 man-years for Bemer's activity.

Another interesting aspect of this film is that it was obviously "shot" at one sitting (or standing), and no attempts were made to edit any abberations from the presentation. Thus a false start at presenting a problem for solution is somewhat amusing:

"... The Indians bought Manhattan island at a cost of $24. [Pregant Pause] Pardon me, the Indians sold Manhattan Island at a cost of $24 ..."

The development of a program, its punching onto cards and the compilation process are much as one would expect today in a batch environment. The film shows some shots of the IBM 704 flashing its lights during compilation and the output coming out of the printer at 100 lines per minute!

FIRST TEXTBOOKS

While there were a few textbooks that contained a chapter on programming languages [20] the production of a single topic textbook on FORTRAN was perhaps inhibited by the excellence of the user's manuals produced by the Backus group. In early 1961, Elliott Organick, then at the University of Houston produced an "internal" booklet on FORTRAN which was marketed through the

---

[20] See for example: Andree, R.V., Programming the IBM 650 ..., Henry Holt and Co., New York, 1958, Ch.8, "Compilers".

university bookstore for local use. This was accompanied later in the same year by another volume of drill exercises. However by this time Daniel McCracken had overcome the opposition from a commercial publisher and produced the first textbook solely devoted to FORTRAN [21]. Surprisingly enough this volume was not much larger than the original user's manual and maintained the wide margins and clear text of that earlier IBM manual. The review in Computing Reviews was not encouraging:

> (Computing Reviews, Vol.3, No.1, Rev. 1421, 1962 January, p. 22) states: "There are versions of FORTRAN for the IBM 650, 1620, 704, 709, 7090, and for the Honeywell 800, the Philco ALTAC, and the Control Data 1604. Since each version has its own description this latest work might seem redundant but it does have some definite advantages."

It is interesting to note that five years later Computing Reviews refrained from soliciting formal reviews of FORTRAN texts due to their "proliferation" and resorted instead to merely publishing an extract from the author's introduction!

LOAD AND GO SYSTEMS

Although Bemer had invented load-and-go systems for the PRINT 1 language, their emergence as a "standard" implementation of a programming language did not occur until the early 1960's.

IBM 1620

The IBM 1620 was the first (IBM) machine which provided the user

---

[21] McCracken, D.D., A Guide to FORTRAN Programming, John Wiley and Sons, New York NY, 1961, 88pp.

with a truly interactive capability and a machine language which was much better human engineered than its competitors such as the (Royal McBee) LGP-30 or the (Bendix) G-15. Thus users could benefit from the concepts of compiling programs and immediately executing those programs from memory without resorting to the production of an object deck or the bother of reloading the compiled program.

FORGO -- University of Wisconsin

Part of the impetus for this movement was again the idea that "anything IBM can do, a user can do better." Thus to save time in an open shop environment with a multitude of engineering students desiring to compile and run programs, Charles Davidson at the University of Wisconsin implemented a FORTRAN II in this load-and-go environment named FORGO. It was the first of many other similar systems for the 1620 which included systems from the Air Force Institute of Technology (AFIT FORTRAN by Richard Pratt), UT FORTRAN (from the University of Toronto) and KINGSTRAN (developed by a joint team from the University of Toronto, Dupont of Canada, Ltd., and Queen's University at Kingston). This lineage eventually led to the development of WATFOR for the IBM 360 and the plethora of similar systems developed by universities for FORTRAN and other languages.

WHAT IS WRONG WITH FORTRAN?

FORTRAN is regularly criticized, along with COBOL and other languages of the same era, as being a dinosaur that will not die. While it is true that much of the world's scientific programming

is still being accomplished in FORTRAN, that may be due to the use of the language by other than computer scientists and the lack of high level programming language education in engineering and science schools.

LACK OF STRONG TYPING

One of the criticisms of FORTRAN is the lack of strong typing, which is interpreted as the lack of a requirement that every variable be included in a type-declaration statement.

FORTRAN introduced the concept of name-type relationships

In fact, FORTRAN in its original form was the first to introduce typing based on the syntactic characteristics of the name of an identifier; the problem was (is) that the programmer may not always be as aware of this associativity as is required by the language. If there be an error, the blame should not be heaped on the shoulders of the language originators; it was only in later versions under pressure from users that TYPE statements were added and thus the name-type relationship became a "default" association which is now so much decried.

LACK OF STRUCTURED PROGRAMMING SUPPORT

To attempt to introduce into a language, which reflected the programming habits and practices of the 1950's, the desires and demands of the 1970's is a project which is almost doomed to failure before it is started. The design of the 1977 FORTRAN merely took the style of the language and used that as clothing for another concept while at the same time including all the old features so as to maintain upward compatability.

COMMON/EQUIVALENCE

If there be any "Sins of Programming" the foremost two must be:

Thou shalt not use global variables, and

Thou shalt not use false names.

FORTRAN (IV admittedly) introduced both of these concepts in response to the demands of the time. Programmers wanted to use COMMON and EQUIVALENCE and implementers who wanted their systems to be used introduced them into their language. Following the first stage of FORTRAN development (say up to 1960) it was natural for a language implementer to provide any additional language features he could provided that the cost was minimal. Thus very early in this process of evolution, the subscript restrictions introduced by Backus et al in order to minimize the problems associated with optimization and to force the development of FORTRAN programs which would be optimizable to a degree which was not expected of a programmer, were relaxed and any meaningful expression became acceptable. Even to the point where expressions were meaningless (such as where the result would be a real number) a default conversion procedure was introduced. Thus FORTRAN became:

"... a collection of warts held together by bits of Syntax."

Extensibility -- has reached its limit

FORTRAN has now been extended to a point where it is doubtful whether the originators can recognize it. Of course one can say the same thing about children and their growth process, it is the environment which influences both mental and physical growth;

FORTRAN has now been extended to the point where it is now more like some other language than its original self.

Modifiability -- must still be FORTRAN

To modify FORTRAN any more will mean that the result is no longer FORTRAN; perhaps that is what the 3x3 committee of IBM and SHARE recognized in 1964 when they decided to design a new language rather than introduce FORTRAN VI. And they called it NPL, MPL, MPPL, ..., PL/I.

EASY TO HAVE MEANINGFUL ERRORS

Perhaps one of the difficulties with FORTRAN, and one which could not have been anticipated by its designers, and which is not well understood today even though we try to do something about it, is the problem of being able to develop programs which look correct but which due to some very small abberation are semantically wrong while being syntactically correct.

Venus Probe Problem

Perhaps the most famous example of this in FORTRAN is that pertaining to the first American probe sent to Venus. The probe was lost due to a program fault caused [22] by the inadvertent substitution of a statement of the form

```
DO 3 I = 1.3
```

for one of the form

```
DO 3 I = 1,3
```

---

[22] Horning, J., A Note on Program Reliability, ACM SIGSOFT, Software Engineering Notes, Vol.4, No.4, 1979 Oct., p.6.

which went undetected throughout the "career" of the probe.

POPULARITY

One of the major factors in keeping FORTRAN alive must be its immense popularity outside the computer community; that is, amongst users whose primary vocation is not computing.

1976-77 Hamblin Survey

A survey of institutions of Higher Education showed that 71% were using FORTRAN (i.e. had it on the system) as contrasted with 59% with COBOL, 55% with BASIC and a lowly 9% with Pascal. Since the survey was taken in 1977 then this latter figure may have changed significantly.

1980 GUIDE Questionnaire

A slightly more recent survey, though amongst a group who one would expect to have less interest in a scientific language, provides the following data points:

> Amongst programmers (in GUIDE User's Group installations) who are full time involved in programming, 81% were using COBOL primarily with only 6% being devoted to FORTRAN. Conversely, amongst casual programmers, COBOL only commanded 23% of their usage while FORTRAN had climbed to 18%.

HOW LONG WILL IT LAST?

Attempting to judge the longevity of any programming language is likely to be a slightly fruitless occupation, except in the case of a well established language such as FORTRAN. The ANSI committee responsible for FORTRAN has decided that the next

version of the standard (the last was published in 1978 and refers to the language which is generally known as FORTRAN 77) will contain list of items to be deleted. The purpose of this list will be to warn programmers against using certain language features which are now considered to be obsolete or unappropriate in today's programming environment. If the standard is published in the same period as the previous two, then this list should be available in 1988 and the succeeding standard (which will not contain those items) will be ready in 1999. Thus FORTRAN will exist in basically its current form until the end of the twentieth century. Perhaps Tony Hoare [23] expressed this longevity best:

> "I don't know what the language of the year 2000 will look like, but I know it will be called FORTRAN."

---

[23] Personal Conversation following the Turing Lecture at the Annual ACM Conference in Nshville TN, 1980 Nov.

```
****B   END  JOB  202 B1027   F25    8.32.20 AM  10 NOV 82  ORIGIN VM1  F25  OUTPUT MVS1  R48.PU1  B****
****B   END  JOB  202 B1027   F25    8.32.20 AM  10 NOV 82  ORIGIN VM1  F25  OUTPUT MVS1  R48.PU1  B****
****B   END  JOB  202 B1027   F25    8.32.20 AM  10 NOV 82  ORIGIN VM1  F25  OUTPUT MVS1  R48.PU1  B****
****B   END  JOB  202 B1027   F25    8.32.20 AM  10 NOV 82  ORIGIN VM1  F25  OUTPUT MVS1  R48.PU1  B****
```