

## TABLE OF CONTENTS

- I. GENERAL INTRODUCTION
  - A. Definition of high-level programming language
  - B. Basic programming language terminology's
  - C. Advantages and Disadvantages of high-level programming
  
- II. HISTORY OF FORTRAN
  - A. Biography of John Backus
  - B. Origins of FORTRAN
  - C. Construction of the compiler
  - D. Rivals and Descendants of FORTRAN
  
- III. VERSIONS OF FORTRAN
  - A. FORTRAN II
  - B. FORTRAN III
  - C. FORTRAN IV
  
- IV. SIGNIFICANT CONTRIBUTIONS TO TECHNOLOGY
  
- V. SIGNIFICANT EXTENSIONS OF FORTRAN
  
- VI. INFLUENCE OF FORTRAN
  
- VII. FORTRAN'S SUCCESS
  
- VIII. CONTINUED EVOLUTION OF FORTRAN
  
- IX. CONCLUSION

## **GENERAL INTRODUCTION**

Programming languages has developed to be the most important means of communication between the person with a problem and the digital computer used to solve it. If the computer had to be instructed in machine language, it would be unrealistic to find a solution to most problems. This is because most machines operate in binary; therefore the only means of communication between the user and the computer itself is the programming language.

### **A. Definition of Programming Language**

According to Sammet, author of *Programming Languages: History and Fundamentals*, "a programming language is a set of characters with rules for combining them. It has the following characteristics; 1) Machine code knowledge is unnecessary, 2) Potential for conversion to other computers, 3) Instruction explosion, and 4) problem-oriented notation."

### **B. Basic Programming Language Terminology's**

1. Source Program: This is the actual program written in a higher-level language and it is put into the computer mainly to obtain results.
2. Object Program: This program can exist in binary form or in a somewhat complex symbolic assembly language form. It is frequently used to signify the outcome of translating the source programming to an assembly level.

3. **Compiler:** This is a program that interprets a source program written in a certain programming language to an object program that is capable of running on a certain computer. The compiler should be able to perform the following functions: Examination of the source code, recouping of suitable subroutines from a library, allocates storage, and developing of real machine code.
4. **Interpreter:** This is a program that executes a source program. The result is an actual answer.
5. **Automatic Coding:** This is the process of writing the source program and translating it to a form that can be run on a computer.
6. **Automatic Programming:** Automatic coding is a specific subset of automatic programming. Automatic programming is the process used by a computer to carry out part of the work involved in the preparation of a program.

### **C. Advantages and Disadvantages of High-Level Programming Language**

#### **Advantages**

- ✓ The most important advantage of high-level programming is that it is easy to learn as compared to machine-oriented language. There are two aspects to this; First, even though the programming language can be complicated, its ease of learning comes about because the notation is fairly related to the

problem zone than it is to the machine code, and second, more focus is placed on the language and the logic of the program, whereas when dealing in machine code the focus is on the characteristic of the physical hardware.

- ✓ The actual coded program is easier to write since the notation is more problem-oriented. The program is also easier to understand once it is written.
- ✓ It is easier to debug a program written using high-level programming language than a program written with low-level language. This is because more attention is placed on the logic of the program and less attention to the details of the machine code. For instance, even though more characters are used in writing READ NEXT RECORD FROM THE TAPE ALPHA than in REDABC, ALPHA, it is difficult to understand the latter.
- ✓ Because of the notation advantages, high-level programming language provides specific documentation automatically. It is also easier to maintain as compared to low-level programming language. There are very few programs that last a long time without requiring a change.
- ✓ The potential for conversion to other computers is considered a major advantage of high-level programming. Conversion is a key problem because programming costs are equal to or even surpasses hardware costs; hence numerous companies have been

unable to purchase new computers. Nevertheless, given that high-level languages are somewhat machine independent, the ease of conversion is a very significant advantage.

- ✓ High-level languages decrease the total amount of elapsed time from the beginning of a problem to its solution. This is specifically true for problems in which a small number of cases need to be run. The elapse time is reduced from months to weeks in some cases or even days to hours in different cases.

### **Disadvantages**

- ✓ The advantages mentioned above do not always exist in some cases. This would result in a comparison between a complicated and strong high-level language and a simple low-level language. Therefore, the high-level language might be very difficult and hard to learn; and also if appropriate attention is placed on the compiler and other aspects of the system, the other advantages may not accumulate. Luckily this rarely happens.
- ✓ With the use of higher-level language, the most evident disadvantage is that the additional process of compilation needs more machine time than the straight assembly process. One specific disadvantage on one-shot problems is that the compilation time occasionally from time to time exceeds the time required to produce the answers. Another disadvantage is

that is the need to recompile every time there is a change in the source program.

- ✓ The compiler sometimes produces inefficient codes. This problem is usually blamed on the compiler unfairly. The problem occurs when the source program is written inefficiently in the higher-level language and as a result inefficient object programs are produced. Even though it is easier to code in higher language than in lower level language there is still a difference between good and bad coding. Thus, no matter how good a compiler is a program written inefficiently in any programming language will produce inefficient object codes.
  
- ✓ It may be difficult to debug a higher-level language as compared to a machine language if the person does not know the compiler does not provide machine code and the proper diagnostics and debugging tools. Therefore if the compiler does not provide proper attention to this feature then the advantages of higher-level languages may be reduced to a great extent.

## **HISTORY OF FORTRAN**

### **A. Biography of John Backus**

John Backus was the inventor of FORTRAN; the first high-level programming language and the most used programming language of physical science. The development of FORTRAN revolutionized

the technology industry and served as a foundation for many generations of languages to come.

John Backus grew up in Wilmington, Delaware, although he was born in Philadelphia in 1924. Backus family was wealthy; he attended Hill School in Pottstown, Pennsylvania. In 1942, he graduated from Hill School and enrolled in University of Virginia. Backus father wanted him to study chemistry. His father was a chemist at one time. Even though Backus disliked lab work, he liked the theoretical part of the science. Backus was expelled after his class attendance fell to once a week. In 1942 he joined the army.

While serving as a corporal in the army in charge of an anti-aircraft crew at Fort Stewart, Georgia, he took an aptitude test that changed his career. He then decided to enroll in a pre-engineering program at the University of Pittsburgh. He took another aptitude test for medical skill and he again enrolled at Haverford College to study medicine. As part of the premed program he worked at Atlantic City hospital. Unfortunately at that time he was diagnosed with brain tumor and underwent an operation in which a plate was installed in his head.

Backus enrolled in Flower and Fifth Avenue Medical school and after nine months he decided that medicine is not for him. Since he liked music he decided to enroll at a radio technician's school. At the school Backus, assisted an instructor perform mathematical calculations for an amplifier curve. Although the work was tedious, it made Backus realize that he had an aptitude

and an interest in mathematics. He then decided to attend Columbia University to study mathematics. During spring of 1949, Backus visited the IBM Computer Center on Madison avenue, where he toured one of IBM's early electronic computers, the Selective Sequence Electronic Calculator (SSEC).

Backus was hired to work on SSEC. With the SSEC, programs had to be entered on punched tape paper because it had no memory for software storage. It was also very slow and undependable. Backus's responsibility was to fix it when there was a problem. Backus invented a program called Speedcoding while he worked on the SSEC. It was the first program that included a scaling factor, which permitted small and large numbers to be stored and manipulated.

Backus wrote a memo to his boss in 1953 that summarized the design of a programming language for IBM's new computer, the 704. The 704 had a floating point, and an indexer, which decreased operating time. He wanted to invent a program that was easy and fast to use while working on the machine. Backus's proposal was approved and a team of programmers and mathematicians were hired to work with him, thus the birth of FORTRAN.

Designed for mathematicians and scientists, FORTRAN is still in use forty years after its introduction. It permits people to work with their computers without an understanding of how the computer works and also learning the machine assembly language.

Backus invented a notation called the Backus-Naur Form, which explains grammatical rules for high-level languages. It is also used in other languages. He also invented the function-level language.

Backus was appointed an IBM Fellow in 1963. He was awarded the W.W. McDowell Award of the IEEE in 1967, National Medal of Science of 1975, the Draper Prize in 1993, and in 1977 the Turing Award of the ACM. In 1991, Backus retired from the computer industry.

On October 28, 1988, John Backus died at the age of 77 in the UCLA Medical Center.

## **B. Origins of FORTRAN**

### **Early Background and Environment**

Prior to 1954, almost all programming was done in machine language or assembly language. The programmer's main effort was dedicated to overcoming the difficulties created by the computers at the time. The computers lacked index registers, built-in floating point, limited instruction sets, and ancient input-output arrangements. Because of the nature of the computers at the time, the services which "automatic programming" rendered to the programmer were anxious to overcome the machine's limitation. Therefore the main concern of some "automatic programming" systems was to permit the use of symbolic addresses and decimal numbers.

Because all the early "automatic programming" systems slowed down the machines, they were costly to use. The reason why they were slow is that they spent most of their time in floating point subroutines. The programmer's experience with slow "automatic programming" systems and problems of putting loops in order and address modification, persuaded them that efficient programming could not be automated. Another reason why the computing community did not take "automatic programming seriously was that "their "automatic programming" systems had almost human abilities to understand the language and the needs of the user; whereas closer inspection of these same systems would often reveal a complex, exception-ridden performer of clerical tasks which was both difficult to use and inefficient." (Wexelblat 26)

In general it was hard to get across to a reader in the late seventies the strength of the uncertainty of "automatic programming" and also about its capability of producing efficient programs, as it was in 1954.

Economics of programming in 1954 was another factor that influenced the evolution of FORTRAN. The cost of the computer was relatively the same as the cost of the programmers associated with the computer center. Also about half of the computer's time was spent debugging. Therefore debugging and programming took up most of the cost of operating a computer. And as the price of the computers dropped the situation became worse. This factor is what led John Backus to propose the FORTRAN project in a memo to his boss at the time Cuthbert Hurd in 1953. Backus stated in the

paper he wrote, *History of FORTRAN I, II, and III* that "I believe that the economic need for a system like FORTRAN was one reason why IBM and my successive bosses, Hurd, Charles DeCarlo and John McPherson, provided for our constantly expanding needs over the next five years without ever asking us to project or justify those needs in a formal budget."

### **Early Stages of the FORTRAN Project**

After the approval of the proposal written by John Backus to his boss at the time, Cuthbert Hurd, to create a realistic automatic programming for 704, Irving Ziller was assigned to the project. They began work in one of the small offices in IBM headquarters at 590 Madison Avenue in New York. John Backus, Harlan Herrick and Irving Ziller developed most of the FORTRAN language. Roy Nutt, who was at the time not a member of the FORTRAN PROJECT, and was also an employee of United Aircraft Corp., designed the input-output language and facilities.

The main goal of the project was to design a language that would make it possible for engineers and scientists to write programs by themselves for the 704.

They formed the "Programming Research group by the fall of 1954, and John Backus was the manager. By November of that year they produced a report the "PRELIMINARY REPORT, Specifications for the IBM mathematical FORMular TRANslating System, FORTRAN", which was dated November 10, 1954 was the earliest important document the exists. The Programming Research Group, Applied

Science Division, of IBM, issued it. According to Sammet " the first sentence of this report states that the IBM Mathematical Formular Translating System or briefly, FORTRAN, will comprise a large set of programs to enable the IBM 704 to accept a concise formulation of a problem in terms of a mathematical notation and to produce automatically a high-speed 704 program for the solution of the problem" (Sammet 143).

In the first paragraph of the report states that "systems which have sought to reduce the job coding and debugging problems have offered the choice of easy coding and slow execution or laborious coding and fast execution." They also proposed that programs " will be executed in about the same time that would be required had the problem been laboriously hand coded." They also stated that " FORTRAN may apply complex, lengthy techniques in coding a problem which the human coder would neither the time nor the inclination to derive or apply" (Wexelblat 30).

In addition the report also stated that " each future IBM calculator should have a system similar to FORTRAN accompanying it. It is felt that FORTRAN offers as convenient a language for stating problems for machine solution as is not known.....After an hour course in FORTRAN notation, the average programmer can fully understand the steps of a procedure stated in FORTRAN language without any additional comments."

The FORTRAN language explained in the "Preliminary Report" had function names of more than three characters, one or two character variables, recurring expressions, arithmetic formulas

and "DO-formulas". Expressions in arithmetic formulas included both integers and floating point quantities

In the *Programmer's Reference Manual* dated October 15, 1956 explained the FORTRAN language in a slightly different way from that of the "Preliminary Report". This was because at the time the "Preliminary Report" was written the authors were not aware of the problems that they would come across later while producing the compiler. There also a few noteworthy deletions such as the Relabel and Relative Constant statements, and inequalities from IF statements. Other changes included; the simplification of the DO statements, increased length of variables to six characters, general enhancement of input-output statements, and addition of FORMAT, CONTINUE, and assignment of GOTO statements.

After the completion of the "Preliminary Report" in late 1954 and early 1955, Harlan Herrick, Irving Ziller, and John Backus gave talks about the plans of FORTRAN to various groups of IBM customers who had purchased the 704.

#### SAMPLE PROGRAM - FORTRAN

**Problem:** Construct a subroutine with parameters A and B such that A and B are integers and  $2 < A < B$ . For every odd integer K with  $A < K < B$ , compute  $f(K) = (3K + \sin(K))^{1/2}$  if K is not prime. For each K, print K, the value of  $f(K)$ , and the word PRIME or NONPRIME as the case may be.

Assume there exists a subroutine or sanction PRIME (K), which determines where or not K is a prime, and assume that library routines for square root, sine and cosine are available.

Program:

```
SUBROUTINE PROBLEM (A,B)
INTEGER A, B
J = 2*(A/2) + 1
DO 10 K = J, B, 2
T = K
IF (PRIME (K) . EQ. 1) GO TO 2
```

```

        E = SQRT (4.*T + COS(T))
        WRITE (1,5) K, E
        GO TO 10
2       E = SQRT (3.*T + SIN(T))
        WRITE (1,6) K, E
10      CONTINUE
5       FORMAT (16, F8.2, 4X, 8H NONPRIME)
6       FORMAT (16, F8.2, 4X, 5H PRIME)
        RETURN
        END

```

*Source: Programming Languages: History and Fundamentals, 151.*

The following are the technical features of FORTRAN:

1. Character set is comprised of the twenty capital letters, ten digits and ten symbols, that is, + - \* / ( ) = .
2. Data name is comprised of a letter followed by zero to four alphanumeric characters. Statement labels have one to four digits. Any string of characters can be used as a data name because there are no reserved words.
3. Language has no delimiters, and the only punctuation is a comma, which is mainly used to separate lists of items. The only operators are the five arithmetic ones. Blanks are not important.
4. A single statement is the smallest executable unit. The DO statements and the tests in an IF statement are handled by the loops.
5. The four categories of procedures defined in FORTRAN are Statements, intrinsic, external functions, and external subroutines.

6. The variable type is determined by its name which begins with one of the following letters I, J, K, L, M, or N, this denotes the INTEGER type whereas all the rest denotes type REAL
7. Integer and floating point is the only arithmetic done. It is not allowed to use real and integer variables or even constants in the same expression.
8. The form  $v = e$ , where  $v$  is the variable name and  $e$  is an arithmetic expression, is the only assignment statement.

### **C. Construction of the compiler**

In early 1955, the FORTRAN compiler was started. It was the ancestor of all modern compilers, and it was the first to have such power and breadth. It took about 25 man-years to produce the first version (IBM [1954]). The initial versions of the FORTRAN compiler of 1955 were roughly as efficient as the assemblers of the time.

### **D. Rivals and Descendants of FORTRAN**

There were other high level languages developed at the time, FORTRAN was not the only one. During the period 1952 - 1957, numerous compiled languages emerged. These languages had an intent was to ease the problems of handling mathematical expressions. Examples of these languages are MATHMATIC, also called AT3, and Internal Translator (IT).

In 1957, a group led by Grace Hopper implemented MATHMATIC on the UNIVAC I. Even though it had remarkable features, that language had very little success. One of its major defects is that it was written for a machine that had no index registers or built-in floating-point arithmetic.

IT got its name from a group that was led by Al Perlis at Carnegie Mellon University. This language was written for the IBM 650 for the sole purpose of simplifying the process of communicating algorithms to the machine. It illustrated that a simple language's compiler can be written quickly if programmers with extraordinary ability do the work and that complete documentation is not needed. IT initially did a syntax analysis and then decoded the source program into the assembly language. Before IT gave way to FORTRAN, it was very famous with users of the IBM 650.

There were many descendants of FORTRAN. One descendant was the PAF (Programmteur Automatique de Formules), which D. Starynkewitch for the SEA invented in France in 1958 machine CAB 500. The main purpose of this language was "to write in plain language, closely similar to spoken French, the instructions to be obeyed in the course of the calculation. Instructions like POSER (PUT) or CALCULER (COMPUTE) or SI A > B ALLER EN (IF B GO TO) were included in the language." (Sammet 165)

In France, PAF made a significant impression. This is mainly because its statements were in French, thus everyone could write, for instance, IMPRIMER AVEC 2 DEC(imales) RC(racine

carree) (de) N - PRINT SQUARE ROOT of N to 2 DECIMALS. At the time any programming language for France had to use French words. FORTRAN instructions were translated into French, and IBM sold two versions, English and French. The users preferred the English form because the whole international scientific community could easily understand a scientific program. It was also more easily transferable from one center to another. In the mid-1960s the terms used in the high-level programming languages were mostly in English.

PAF had a famous descendant; BASIC produced in 1965, after it disappeared with CAB 500. Although the authors of BASIC were not aware of PAF, BASIC had majority of its characteristics.

## **VERSION OF FORTRAN**

### **FORTRAN II**

FORTRAN I had a number of evident defects, and the lack of any automatic process for checking syntax errors by the programmer made it even worse. FORTRAN II was the solution to this problem. Backus, Nelson and Ziller started to plan the correcting these problems in the fall of 1957. A document (Proposed Specification) dated September 25, 1957 characterized the changes as "(a) a need to for better diagnostics, clearer comments about the nature of source program errors, and (b) the need for subroutine definition capabilities." The title of the document is "Proposed Specifications for FORTRAN II for the 704"

and it described a more diagnostic system, the new subroutine definitions and END statements. It also described "how symbolic information is retained in the relocatable binary form of a subroutine so that the "binary symbolic subroutine [BSS] loader" can be implemented references to separately compiled subroutines, and also new prologues for these subroutines and points out that mixtures of FORTRAN-coded and assembly-coded relocatable binary programs could be loaded and run together.

Many changes were made to FORTRAN I. These changes included: (1) some characteristics were deleted which made it hard to translate into machine language, (2) enhancement of the input/output statements, (3) GO TO instruction was extended with the "compute GO TO, and lastly (4) increase of the variable characters to six.

In spring of 1958, FORTRAN II was distributed.

As an illustration of programming in FORTRAN II, think about the summation of a 100 numbers, the numbers to be put into the machine from one of it's input devices and the total to be printed on an output device. The program is as follows:

```
DIMENSION A(100)
READ 2, A
SUM = 0
DO 7 I = 1, 100
7 SUM = SUM + A(I)
PRINT 7, 1, SUM
STOP
1 FORMAT (F 10,4)
2 FORMAT (E 10,3)
END
```

### **FORTRAN III**

During the development of FORTRAN II, Ziller was developing and even more advanced version. It permitted a programmer to write intermixed symbolic instructions and FORTRAN statements. "The symbolic (704) statements could have FORTRAN variables as "addresses"." Another feature is its machine dependency, which consisted of early versions of a number of enhancements that later appeared in FORTRAN IV. Other feature of FORTRAN II included; Boolean expressions, function and routine names, and ability to handle alphanumeric data such as a new FORMAT code "A" similar to codes "I" and "E".

FORTRAN III was never distributed. In the winter of 1958-1959 it was available on a limited scale until early sixties.

### **FORTRAN IV**

FORTRAN IV was developed in 1962. It is still a significant dialect. It is also widely used. The following are significant characteristics of FORTRAN IV:

1. It can explain a variety of algorithms, even though it is used primarily for scientific and engineering computations.
2. Grammar of the language is defined specifically unlike English. For this reason FORTRAN IV algorithms means precisely the same thing to every other person who reads it.

3. Since the language avoids any reference to special devices, numerous types of digital computers can be programmed to acknowledge algorithms written in FORTRAN IV.

### **SIGNIFICANT CONTRIBUTIONS TO TECHNOLOGY**

In comparison to any other development, FORTRAN has probably the most important impact on computing. On the other hand, the most noteworthy contributions made by FORTRAN are its usage rather than its technology. Since it was designed very early, it has been improved to do almost anything. The following are most significant technological contributions made by FORTRAN; (1) the invention of a programming language that could be used on any available hardware, (2) the granting of some control over storage allocation to the programmer using the EQUIVALENCE statements, (3) independence of blanks, and lastly (4) its ease of understanding and learning the language.

### **SIGNIFICANT EXTENSIONS OF FORTRAN**

FORTRAN was extended for use in areas that it was not originally intended for. These extensions are either far-reaching in both realistic usage and implications or minor in concept and character. The following are some of the actual language extensions to FORTRAN include Proposal Writing Language, FORMAC (cross-reference only), QUIKTRAN (cross-reference only), GRAF (cross-reference only), and DSL (cross-reference only).

### **Proposal Writing Language**

This is a rather unique extension to FORTRAN created by Carleton, Lego, and Suarez [CT64]. Twelve statements were added to FORTRAN II, as it existed for the IBM 704 in 1959. These statements included; ALPHABET INPUT, RIGHT MARGIN, LEFT MARGIN, TABULATE, SINGLE SPACE, DOUBLE SPACE, RESORE PAPER, ALPHABETIC INSERT, NUMERIC INSERT FORTRAN, PARAGRAPH, END PARAGRAPH, PREPARE PARAGRAPH, and STOP. The FORTRAN statements are all used, apart from the STOP statement. The STOP statement was changed to some extent to control the termination of the proposal writing process. The preprocessor implements the system and it changes all new statements to CALLS and then to suitable subroutines.

Most of the statements are self-explanatory, only a few need explanation. The statements that are not self-explanatory include; (1) ALPHABETIC INPUT makes the computer read from card format by looking for as many alphabetic variables as named in the list, (2) ALPABETIC INSERT gives the names of variables that are inserted into the text, (3) NUMERIC INSERT makes the insertion of the value of the FORTRAN variable stable, (4) PARAGRAPH statements permits the user to define a subprogram; the subprogram is used to produce a single paragraph of text describing individual items for instance motor, (5) PREPARE PARAGRAPH invokes the subprogram and adds it text to the body of the proposal, with the suitable replacements made to the parameters, and finally, (6) the STOP statement starts a completion phase of the proposal writing system.

## **FORMAC**

FORMAC is an important extension of FORTRAN to do formal algebraic manipulation on the computer. It was an extension of FORTRAN IV on the 7090/94; thus all the characteristics related to FORTRAN IV apply to FORMAC.

The essential concepts of FORMAC (FORmula Manipulation Compiler) was invented by Jean Sammet with the help of Robert Tobey in July, 1962 at IBM's Boston Advanced Programming Department. During that time what they wanted was a formal algebraic capability related to an already existing numerical mathematical language, that was, FORTRAN. The fundamental objective of the work was to create a practical system that would perform formal mathematical manipulation running under IBSYS/IBJOB on the IBM 7090/94. In November 1964 FORMAC was released as a Type III program, in other words it was made available to users in IBM.

There are five major contributions of FORMAC to the technology. First and foremost, it initiated the idea of adding this type of ability as a language to a language that is already being used for numerical scientific problems. According to Sammet this is the most important contribution. Second, it illustrated that a practical system can be created to perform algebraic manipulation on the computer and it is easy to learn and solve engineering and mathematical problems. Third, it illustrated that a limiting factor in the problem solving is the amount of storage. Fourth, it is the creation of a practical algorithm for

doing automatic simplification. Lastly, FORMAC assisted people to turn away from numerical analysis and move back to analytical solutions to problems.

### **QUIKTRAN**

QUIKTRAN is an on-line version of FORTRAN. It was developed at the beginning for the IBM 7040. Work on QUIKTRAN began by a group of people with the guidance of John Morrissey. The original purpose was to enhance user-debugging abilities. This purpose in the long run took the form of a dedicated system, FORTRAN, with a strong debugging a terminal control ability added. In mid- 1963, a first version was running.

QUIKTRAN made some significant contributions to technology, for instance, it was the initial on-line system to use standard equipment and also it remained compatible with existing language.

### **GRAF**

GRAF (GRAaphic Additions to FORTRAN) is an extension of FORTRAN to handle graphics. A display variable, a data type, was added to FORTRAN. The value of the variable is a string of orders that have the ability to generate a display when transmitted to the right device. The names of the display variables are similar to the FORTRAN variables.

## **DSL/90**

DSL/90 is an extension of FORTRAN to stimulate blocked diagrams. It was implemented on the IBM 7090/94.

## **INFLUENCE OF FORTRAN**

Even though the computer community looked on FORTRAN with doubt, the fact remains that it still made writing and developing programs so much easier. Despite the fact that the initial FORTRAN had characteristics of the IBM 704, the later versions of FORTRAN could be used in any machine.

FORTRAN II was very successful by 1959 that any manufacturer had to offer a high-level language as good as FORTRAN in order to sell a scientific machine. For this reason, one language was used for many different machines. Thus FORTRAN became the first machine-independent language.

## **FORTRAN'S SUCCESS**

FORTRAN has been very successful. It attained its goal of illustrating that a high-level language can be adequately efficient to be used in production programming. FORTRAN has always been the most greatly optimized programming language.

In almost all application area, FORTRAN can be used effectively. It is quite open to extension and alteration. Because FORTRAN lacks elaborate data structuring methods, this has prevented its effective application to nonnumerical problems.

## CONTINUED EVOLUTION OF FORTRAN

In conclusion, FORTRAN was the first true high-level language. According to Backus, one of its intent was to allow a problem to be stated in briefly in mathematical notations. It was the most striking success in the history of programming.

The history of the evolution of FORTRAN is comparable to the overall evolution of programming John Backus was the leader, he is recognized for inventing what become the most widely used high-level programming language in the world.

Although newer languages introduced higher-level structures such as the **if-then-else** and **while-do**, FORTRAN still remained the most widely used language. As a result of this, numerous preprocessors, for instance, RATFOR, were designed that acknowledged these structured control structures and translated them into FORTRAN. Since these preprocessors permitted programmers to use FORTRAN without giving up the use of the new control features. This gave birth to a new dialect of FORTRAN called FORTRAN 77. It became an American National Standard. The ANSI's (American National Standards Institute) FORTRAN'S committee began work on the successor of FORTRAN 77, however it took twelve years to complete. It was known as FORTRAN 82, 88, and 90 prior to its approval in 1991.

FORTRAN 95 is a minor improvement of FORTRAN 90. Its features include support for exception handling, parameterized types and object-oriented programming.

Plans are on the way for FORTRAN 2000. Hence, FORTRAN is continuing to evolve into the new millenium.