CONF-790402--8

M

FORTRAN - PAST, PRESENT, AND FUTURE

N. H. Marshall
EG&G Idaho Inc.
Idaho Falls, Idaho

Fortran has been widely used both here and abroad for many years, but it has its shortcomings and has fallen under severe criticism. By modern day criteria, Fortran is archaic. It does not lend itself well to modern structured programming philosophies. But Fortran is changing, and it is becoming better. The newly standardized Fortran 77 is a giant step forward. It has improved Fortran's usefulness and will make it easier to write "structured" programs. X3J3, the committee which produced Fortran 77, is already working on future Fortran standards. These promise to be even more modern and more powerful. The future of Fortran looks good and it looks exciting. It is anticipated that Fortran will continue to be widely used for many years to come.

6

[illegible]

FORTRAN - PAST, PRESENT, AND FUTURE

Fortran has been the workhorse of the scientific programming community for many years. Introduced by IBM well over 20 years ago, Fortran became the first programming language to be standardized in March, 1966. Since that time Fortran usage has grown as the usage of computer has grown. It has retained popularity because it is easy to use, it compiles rapidly, and it produces efficient code. The amount of code written in Fortran and the dollar and time investment in Fortran programs is mind boggling. However Fortran's road to success has not been smooth and there have been many rocky times.

FORTRAN AND "STRUCTURED" PROGRAMS

Ten or more years ago people began to realize that computer programming should not be a hapazard process but instead it should be a carefully designed and organized one. Theories of "structured" programming emerged and people began to think in terms of IF-THEN-ELSE and DO-WHILE like constructs. These constructs could be realized in Fortran only with the use of GO TO statements, and the elimination of GO TO statements was the heart of a controversy among computer scientists in the late 1960's. Some felt GO TO's should be eliminated altogether from computer languages, others felt they were necessary, but all agreed that their use should be minimized. Because it is difficult to write "structured" Fortran without propagating GO TO statements, many, especially in the academic world, felt that Fortran should be replaced with a more "structured" language. For a short time the cry was heard that "Fortran is dead". But due to the enormous industrial investment in Fortran and because it compiled quickly producing efficient code, it did not die. Today, very much alive, Fortran is changing to meet the needs of users and is expected to continue to be the industrial workhorse throughout the world. The new Fortran standard contains an IF-THEN-ELSE construct and future Fortrans will contain new looping constructs, so that future Fortran programmers will find it easy to write "structured" programs.

FIRST COMPUTER LANGUAGE TO BE STANDARDIZED

Fortran was first standardized in March 1966 by the then American Standards Association (ASA), by the X3.4.3 subcommittee. In 1969, the name of the American Standards Association was changed to American National Standards Institute (ANSI) and the X3.4.3 subcommittee has subsequently become known as the X3J3 technical committee.

As early as 1967, questions arose about the 1966 Fortran standard and the X3.4.3 committee went to work to clarify for it. Two such reports were published in the Communications of the ACM, one in 1969[1] and one in 1971[2].

In view of the many extensions to the 1966 standard, the X3J3 committee decided in 1970 that it would be more productive to abandon the clarification work and devote their time to a revision. Since that time, this committee has met about six times a year with many members devoting almost full time to the work. Hundreds of proposals have been considered. Correspondence from all over the nation and the world was reviewed at each meeting. X3J3 has tried to be responsive to the needs of the computing industry. Because of this and because of the magnitude of the work, the revision effort was frustratingly slow. With decisions as to what features should be included, what was the best syntax, and with many hours of editing, the proposed new standard consumed much time and money. Their effort produced Fortran 77, which became a new Fortran standard on April 3, 1978. It is estimated that this effort cost over two million dollars. The 1966 Fortran standard contained 26 pages plus appendicies, the 1978 standard contains approximately 165 pages plus appendicies. The increase in bulk was due not only to the expansion of Fortran features but also because X3J3 made an effort to produce a document which was more readable and usable than the 1966 standard.

FORTRAN 77

The new Fortran standard is entitled "American National Standard Programming Language Fortran, ANSI X3.9-1978" and is published by the American National Standards Institute. There is an excellent article entitled "Fortran 77" in the communications of the ACM[3] which discusses the new standard and its features. The following quote regarding the criteria which guided the X3J3 committee is taken from that article.

> "The main criteria followed by the committee while developing the new standard were:
>
> 1. Inclusion of new features whose utility has been proven by actual usage.
> 2. Inclusion of new features that make programs easier to transport from one processor to another.
> 3. Minimal increase in the complexity of the language or of processors for the language.
> 4. Avoidance of features that conflict with X3.9-1966.
> 5. Elimination of features in the 1966 standard only if there is a clearly demonstrated reason for doing so.
> 6. Production of a more precise description of the language."

Features of Fortran 77 which are extensions from the 1966 standard have been widely publicized and will not be recounted in detail here; however, comments on some of them which augment the 1966 Fortran do seem appropriate.

One of the big additions to Fortran is that of character data type. Character manipulation in Fortran has always been difficult, and since there has been no character type, it has been done "under the guise of" another data type, usually real or integer. Included also in Fortran 77 is a concatenation operator and substring operations so that character and text manipulation may be done easily. Fortran 77 also offers some other powerful features related to character data type which may not be so obvious to the casual reader. Some of these are:

a) The use of character constants and character variables in decisions. Not only can two character strings be tested for equality, but Fortran 77 defines a partial collating sequence so that it is meaningful to ask if one character string is greater than or less than another.

b) It will be possible to use character constants or character variables as format identifiers in input/output statements. This feature will make it easy to select from one or more I/O format definitions at execution time.

c) It will be easy to modify I/O format definitions at execution time. By using the substring operations with character variables used as format identifiers, they may be easily modified during execution.

d) Expressions will be permitted in output statements. Since this includes character expressions, text to be printed may be included in the output statement output list if desired. In the 1966 Fortran it was restricted to the Format statement.

e) The ability to do internal I/O, i.e., the ability to read from memory or write to memory, is provided in an easy and natural way in Fortran 77. (This is the same capability provided by the ENCODE and DECODE statement in some Fortran extensions.) This is done by using an appropriate character variable as the unit identifier in an input or output statement. The I/O operation causes data to be moved between that character variable and the variables in the input/output list with any data conversions controlled by the FORMAT identifier.

Another new feature in Fortran 77 which will do much to enhance it to programmers is the addition of an IF-THEN-ELSE construct. This will permit programmers to write "structured" Fortran without the use of excessive GO TO statements. This feature will change the style of Fortran programmers.

Fortran 77 also includes extensions to the DO loop mechanism, new format descriptors, direct access I/O, the PARAMETER statement, generic functions, and many more. These make the prospect of using Fortran 77 exciting.

Fortran 77 is here. While Fortran 77 compilers are still not generally available, they undoubtedly will be announced over the next year or two. It will be an interesting transition.

FUTURE FORTRANS -- WHAT WILL THEY BE LIKE?

But what of the future? Where is Fortran going? When will the next Fortran standard be issued? These questions obviously cannot be answered exactly, but it is interesting to speculate. The ANSI X3J3 technical committee certainly does not regard Fortran 77 as the ultimate language. It is more of an intermediate standard to replace the 1966 standard which was so out of date. As Fortran 77 was completed, X3J3 did not even break its stride. Not a meeting was missed. Groups had already been organized to investigate the need of future standards before Fortran 77 was finalized. The X3J3 is currently looking to 1983-84 for the next Fortran standard. Whether or not this is realistic remains to be seen, certainly the amount of work they have outlined for themselves is overwhelming.

The X3J3 is currently looking to a "core plus modules" approach for future Fortrans. The core Fortran would be a complete language with essentially all the functionality of Fortran 77. Modules could then be defined which would interface with the core and provide desirable extensions to the core Fortran. This may be kind of a radical approach, but currently it seems to have merit. There are three motivations for this approach.

1) This would permit collateral standards areas such as data base management, real time process control, and graphics to be developed. Such standards would form modules which would interface with the core Fortran.

2) This would permit a special features module which could contain features not in the core Fortran, but which should be included in a Fortran standard. One such special feature could be array processing.

3) This would permit an "old features" module. There are archaic features in current Fortran which both the public and the X3J3 committee feels should be deleted, but which cannot be because of the enormous number of working programs which contain them. By placing these features in a separate module, they will be preserved for those who need them, and they will not inhibit the development of better techniques which provide the same functionality.

If the "core plus module" approach is adopted, it is the intent of the X3J3 committee that the number of modules remain small. The key to this approach is the interface between the core and the modules. For this reason, the committee is investigating ways of enhancing the procedure calling mechanisms in Fortran.

Future Fortrans will undoubtedly include some new looping constructs. It appears likely that these will include both a form of an unconditional looping construct as well as a conditional one. The unconditional looping construct will cause the body of the loop to be executed repeatedly until some kind of an exit statement is executed. The conditional looping construct will permit the body of the loop to be executed until some looping condition is satisfied. Note that conditional looping construct will include the functionality of the DO WHILE construct.

Another candidate for inclusion in future Fortrans is some form of a CASE construct. This will permit design of blocks of Fortran statements which may be selected and executed independently depending upon some initial condition.

Future Fortrans may also contain internal procedures. One of the principle issues here is the scope of names in internal procedures.

Some find of array processing instructions will proabably appear in future Fortrans. This seems particularly desirable since future hardware will undoubtedly support some form of vector manipulations.

Another data type which may appear in future Fortrans is BIT data type. People still have a need for "bit-twiddling" and the omission of some kind of bit operations in Fortran 77 was felt by some to be a serious oversight.

Future Fortrans may also have a "new look". The X3J3 committee is investigating such features as:

1) Free-form of Fortran statements. This would delete old Fortran conventions such as a "C" in position one for comments, a nonzero, nonblank character in position six for continuation lines, and the requirement that statements start in position seven or after.

2) Longer variable names. Many people feel that six characters are inadequate for writing meaningful mnemonic names.

3) In-line comments. Many people feel the need to write comments on the same line as the Fortran statements.

4) Multiple statements per line. Many other languages support this feature and some people feel that it is a desirable feature.

The computer science field is rapidly changing and computer languages must change too, or they soon become obsolete. Fortran is not dead, it is very much alive and viable. It is growing and maturing to stay current with the programming needs. Fortran expects to maintain its position of workhorse of the scientific programming community.

REFERENCES

1 Clarification of Fortran Standards -- initial progress Comm ACM 12, 5 (May 1969), p. 289-294.

2 ANSI Subcommittee X3J3, Clarification of Fortran Standards -- second report. Comm ACM 14, 10 (Oct 1971), p. 628-642

3 Fortran 77, Comm ACM 21, 10 (Oct 1978), p. 806-820.

FORTRAN - PAST, PRESENT, AND FUTURE

Fortran has been the workhorse of the scientific programming community for many years. Introduced by IBM well over 20 years ago, Fortran became the first programming language to be standardized in March, 1966. Since that time Fortran usage has grown as the usage of computer has grown. It has retained popularity because it is easy to use, it compiles rapidly, and it produces efficient code. The amount of code written in Fortran and the dollar and time investment in Fortran programs is mind boggling. However Fortran's road to success has not been smooth and there have been many rocky times.

FORTRAN AND "STRUCTURED" PROGRAMS

Ten or more years ago people began to realize that computer programming should not be a hapazard process but instead it should be a carefully designed and organized one. Theories of "structured" programming emerged and people began to think in terms of IF-THEN-ELSE and DO-WHILE like constructs. These constructs could be realized in Fortran only with the use of GO TO statements, and the elimination of GO TO statements was the heart of a controversy among computer scientists in the late 1960's. Some felt GO TO's should be eliminated altogether from computer languages, others felt they were necessary, but all agreed that their use should be minimized. Because it is difficult to write "structured" Fortran without propagating GO TO statements, many, especially in the academic world, felt that Fortran should be replaced with a more "structured" language. For a short time the cry was heard that "Fortran is dead". But due to the enormous industrial investment in Fortran and because it compiled quickly producing efficient code, it did not die. Today, very much alive, Fortran is changing to meet the needs of users and is expected to continue to be the industrial workhorse throughout the world. The new Fortran standard contains an IF-THEN-ELSE construct and future Fortrans will contain new looping constructs, so that future Fortran programmers will find it easy to write "structured" programs.

FIRST COMPUTER LANGUAGE TO BE STANDARDIZED

Fortran was first standardized in March 1966 by the then American Standards Association (ASA), by the X3.4.3 subcommittee. In 1969, the name of the American Standards Association was changed to American National Standards Institute (ANSI) and the X3.4.3 subcommittee has subsequently become known as the X3J3 technical committee.

As early as 1967, questions arose about the 1966 Fortran standard and the X3.4.3 committee went to work to clarify for it. Two such reports were published in the Communications of the ACM, one in 1969[1] and one in 1971[2].

In view of the many extensions to the 1966 standard, the X3J3 committee decided in 1970 that it would be more productive to abandon the clarification work and devote their time to a revision. Since that time, this committee has met about six times a year with many members devoting almost full time to the work. Hundreds of proposals have been considered. Correspondence from all over the nation and the world was reviewed at each meeting. X3J3 has tried to be responsive to the needs of the computing industry. Because of this and because of the magnitude of the work, the revision effort was frustratingly slow. With decisions as to what features should be included, what was the best syntax, and with many hours of editing, the proposed new standard consumed much time and money. Their effort produced Fortran 77, which became a new Fortran standard on April 3, 1978. It is estimated that this effort cost over two million dollars. The 1966 Fortran standard contained 26 pages plus appendicies, the 1978 standard contains approximately 165 pages plus appendicies. The increase in bulk was due not only to the expansion of Fortran features but also because X3J3 made an effort to produce a document which was more readable and usable than the 1966 standard.

FORTRAN 77

The new Fortran standard is entitled "American National Standard Programming Language Fortran, ANSI X3.9-1978" and is published by the American National Standards Institute. There is an excellent article entitled "Fortran 77" in the communications of the ACM[3] which discusses the new standard and its features. The following quote regarding the criteria which guided the X3J3 committee is taken from that article.

"The main criteria followed by the committee while developing the new standard were:

1. Inclusion of new features whose utility has been proven by actual usage.
2. Inclusion of new features that make programs easier to transport from one processor to another.
3. Minimal increase in the complexity of the language or of processors for the language.
4. Avoidance of features that conflict with X3.9-1966.
5. Elimination of features in the 1966 standard only if there is a clearly demonstrated reason for doing so.
6. Production of a more precise description of the language."

Features of Fortran 77 which are extensions from the 1966 standard have been widely publicized and will not be recounted in detail here; however, comments on some of them which augment the 1966 Fortran do seem appropriate.

One of the big additions to Fortran is that of character data type. Character manipulation in Fortran has always been difficult, and since there has been no character type, it has been done "under the guise of" another data type, usually real or integer. Included also in Fortran 77 is a concatenation operator and substring operations so that character and text manipulation may be done easily. Fortran 77 also offers some other powerful features related to character data type which may not be so obvious to the casual reader. Some of these are:

a) The use of character constants and character variables in decisions. Not only can two character strings be tested for equality, but Fortran 77 defines a partial collating sequence so that it is meaningful to ask if one character string is greater than or less than another.

b) It will be possible to use character constants or character variables as format identifiers in input/output statements. This feature will make it easy to select from one or more I/O format definitions at execution time.

c) It will be easy to modify I/O format definitions at execution time. By using the substring operations with character variables used as format identifiers, they may be easily modified during execution.

d) Expressions will be permitted in output statements. Since this includes character expressions, text to be printed may be included in the output statement output list if desired. In the 1966 Fortran it was restricted to the Format statement.

e) The ability to do internal I/O, i.e., the ability to read from memory or write to memory, is provided in an easy and natural way in Fortran 77. (This is the same capability provided by the ENCODE and DECODE statement in some Fortran extensions.) This is done by using an appropriate character variable as the unit identifier in an input or output statement. The I/O operation causes data to be moved between that character variable and the variables in the input/output list with any data conversions controlled by the FORMAT identifier.

Another new feature in Fortran 77 which will do much to enhance it to programmers is the addition of an IF-THEN-ELSE construct. This will permit programmers to write "structured" Fortran without the use of excessive GO TO statements. This feature will change the style of Fortran programmers.

Fortran 77 also includes extensions to the DO loop mechanism, new format descriptors, direct access I/O, the PARAMETER statement, generic functions, and many more. These make the prospect of using Fortran 77 exciting.

Fortran 77 is here. While Fortran 77 compilers are still not generally available, they undoubtedly will be announced over the next year or two. It will be an interesting transition.

FUTURE FORTRANS -- WHAT WILL THEY BE LIKE?

But what of the future? Where is Fortran going? When will the next Fortran standard be issued? These questions obviously cannot be answered exactly, but it is interesting to speculate. The ANSI X3J3 technical committee certainly does not regard Fortran 77 as the ultimate language. It is more of an intermediate standard to replace the 1966 standard which was so out of date. As Fortran 77 was completed, X3J3 did not even break its stride. Not a meeting was missed. Groups had already been organized to investigate the need of future standards before Fortran 77 was finalized. The X3J3 is currently looking to 1983-84 for the next Fortran standard. Whether or not this is realistic remains to be seen, certainly the amount of work they have outlined for themselves is overwhelming.

The X3J3 is currently looking to a "core plus modules" approach for future Fortrans. The core Fortran would be a complete language with essentially all the functionality of Fortran 77. Modules could then be defined which would interface with the core and provide desirable extensions to the core Fortran. This may be kind of a radical approach, but currently it seems to have merit. There are three motivations for this approach.

1) This would permit collateral standards areas such as data base management, real time process control, and graphics to be developed. Such standards would form modules which would interface with the core Fortran.

2) This would permit a special features module which could contain features not in the core Fortran, but which should be included in a Fortran standard. One such special feature could be array processing.

3) This would permit an "old features" module. There are archaic features in current Fortran which both the public and the X3J3 committee feels should be deleted, but which cannot be because of the enormous number of working programs which contain them. By placing these features in a separate module, they will be preserved for those who need them, and they will not inhibit the development of better techniques which provide the same functionality.

If the "core plus module" approach is adopted, it is the intent of the X3J3 committee that the number of modules remain small. The key to this approach is the interface between the core and the modules. For this reason, the committee is investigating ways of enhancing the procedure calling mechanisms in Fortran.

Future Fortrans will undoubtedly include some new looping constructs. It appears likely that these will include both a form of an unconditional looping construct as well as a conditional one. The unconditional looping construct will cause the body of the loop to be executed repeatedly until some kind of an exit statement is executed. The conditional looping construct will permit the body of the loop to be executed until some looping condition is satisfied. Note that conditional looping construct will include the functionality of the DO WHILE construct.

Another candidate for inclusion in future Fortrans is some form of a CASE construct. This will permit design of blocks of Fortran statements which may be selected and executed independently depending upon some initial condition.

Future Fortrans may also contain internal procedures. One of the principle issues here is the scope of names in internal procedures.

Some find of array processing instructions will proabably appear in future Fortrans. This seems particularly desirable since future hardware will undoubtedly support some form of vector manipulations.

Another data type which may appear in future Fortrans is BIT data type. People still have a need for "bit-twiddling" and the omission of some kind of bit operations in Fortran 77 was felt by some to be a serious oversight.

Future Fortrans may also have a "new look". The X3J3 committee is investigating such features as:

1) Free-form of Fortran statements. This would delete old Fortran conventions such as a "C" in position one for comments, a nonzero, nonblank character in position six for continuation lines, and the requirement that statements start in position seven or after.

2) Longer variable names. Many people feel that six characters are inadequate for writing meaningful mnemonic names.

3) In-line comments. Many people feel the need to write comments on the same line as the Fortran statements.

4) Multiple statements per line. Many other languages support this feature and some people feel that it is a desirable feature.

The computer science field is rapidly changing and computer languages must change too, or they soon become obsolete. Fortran is not dead, it is very much alive and viable. It is growing and maturing to stay current with the programming needs. Fortran expects to maintain its position of workhorse of the scientific programming community.

REFERENCES

[1] Clarification of Fortran Standards -- initial progress Comm ACM 12, 5 (May 1969), p. 289-294.

[2] ANSI Subcommittee X3J3, Clarification of Fortran Standards -- second report. Comm ACM 14, 10 (Oct 1971), p. 628-642

[3] Fortran 77, Comm ACM 21, 10 (Oct 1978), p. 806-820.