What's New in Computers

FORTRAN 95

V Rajaraman

V Rajaraman is with the Jawaharlal Nehru Centre for Advanced Scientific Research, and the Indian Institute of Science, Bangalore. Several generations of scientists and engineers in India have learnt to program using his lucidly written books on Fortran.

This article describes the features of the latest International standard for Fortran known as Fortran 95 which was published in October 1996.

Introduction

The death of Fortran, the oldest high level programming language, has been predicted by many computer scientists but it refuses to die. It is alive and kicking and is steadily improving. The latest version of Fortran, *Fortran 95*, has been published as an International Standard in October 1996. In fact, Fortran has come a long way since its compiler was written in 1954 by John Backus and his group (*Box 1*). Fortran 90 and its successor Fortran 95 have introduced revolutionary changes in Fortran (*Box 2*). It has now absorbed all the good features of programming languages such as Pascal and C and has gone further in providing facilities to compute with arrays and matrices. Fortran 90 is not an evolution but a revolution in the development of Fortran. It has introduced many new features not available in Fortran 77. The most useful of these for numerical computing are:

- Operations on whole arrays which enable Fortran compilers to optimize code generated for pipelined/vector and parallel computers.
- Allocatable arrays which allow good storage management.
- The length of mantissa and the range of exponents of reals may be parametrised by a KIND declaration which allows Fortran 90 code to be machine independent.
- User defined data structures and operators ease programming a variety of applications.
- Pointer datatype allows efficient manipulation of sparse

matrices, lists and graphs.

- Recursive procedures simplify manipulation of recursively defined data structures and writing recursive functions.

We give some code fragments to illustrate the simplicity of dynamic allocation of storage and array assignment in Fortran 90 programs. Comments follow exclamation mark ! in Fortran 90.

```
REAL, DIMENSION (:, :, :), ALLOCATABLE :: D
READ *, N
ALLOCATE D (N,  N + 1, 10 ) !  Allocate storage
A = D ( 1 : 5, 1 : 6, 1 : 4 ) ! Assign part of D to A
B = D ( 6 : 10, 7 : 12, 1 : 4 ) ! Assign part of  D to B
```

Operations on arrays may be performed without using explicit DO loops needed in Fortran 77. For example, we can write:

$$C = A + B$$

where A and B are conformable arrays in Fortran 90. In Fortran 77 this would have been written as:

Box 1

I don't know what the characteristics of the standard language for scientific and engineering computation in the year 2000 will be ..... but I know it will be called Fortran.

John Backus
Designer of the first FORTRAN Compiler

John Backus

Box 2 History of Fortran

1954	Preliminary report - specification for IBM mathematical Formula Translating System, FORTRAN (John Backus and others).
1957	FORTRAN for IBM 704 developed.
1958	FORTRAN II for IBM 704.
1962	FORTRAN IV for IBM 7030 STRETCH Computer.
1966	American Standard Fortran - FORTRAN 66.
1978	American National Standards Institute (ANSI) FORTRAN 77.
1980	International Standards Organization (ISO) adopts ANSI Standard of FORTRAN 77.
1991	ISO publishes Fortran 90 standard.
1992	ANSI adopts ISO standard Fortran 90.
1996	ISO publishes Fortran 95 standard.
2001	The next Fortran standards document scheduled to be released.

```
      DO 100 K = 1, N1
         DO 200 J = 1, N2
            DO 300 I = 1, N3
               C ( I, J, K ) = A ( I, J, K ) + B ( I, J, K)
300         CONTINUE
200      CONTINUE
100   CONTINUE
```

The Fortran 90 array assignment is not only more elegant but also enables the compiler to recognise that the individual element assignments are independent of one another. If this program is

Box 3 New Fortran 90 Features Compared with Fortran 77

- Fortran 77 programs can be run without change using Fortran 90 compilers.
- Fortran 90 programs may be written in a free form and need not adhere to the fixed format imposed in Fortran 77.
- Variable and procedure names may extend upto 31 characters. Underscore is allowed as a character.
- The length of integers and the precision and exponent range of reals need not be fixed. They may be parametrised by a KIND declaration.
- Use of IMPLICIT NONE statement forces all variables to be declared thereby providing better security to programs.
- User defined data types and operators allow a programmer to define data structures and operations on such structures.
- Data structures and procedures may be encapsulated in a MODULE. This allows definition of abstract data types and language extention.
- Packaging data in MODULEs allows controlled global access to entities within the MODULEs. COMMON and labelled COMMON of Fortran 77 need not be used.
- Arrays can be treated as a single object allowing operations on whole arrays without using a DO loop.
- Dynamically allocatable arrays simplifies writing of procedures with arrays.
- Simplified DO loop syntax allows writing error free programs with loops.
- CASE statement allows writing a more readable code.
- Recursive procedures are allowed in Fortran 90.
- Specification of input arguments of procedures as INTENT (IN) allows the compiler to prevent accidental change of these variables.
- Optional arguments in procedures provide greater flexibility in writing programs.
- Fortran 90 has a POINTER data type. This allows creation and manipulation of dynamic data

executed on a parallel computer, the compiler will recognise the independence between iterations and exploit it to optimize the code generated for it.

There are several new intrinsic functions in Fortran 90 which simplify operations with arrays. For example, one such function is to multiply matrices. One may write:

$$C = MATMUL (A, B)$$

to multiply two conformable matrices A and B to get the product matrix C.

We have given in *Box 3* a more complete list of the features in Fortran 90 not available in Fortran 77. Due to large investments made in developing Fortran 77 programs upward compatibility was maintained in Fortran 90 standard (see *Box 4*).

Fortran and Parallel Computers

When Fortran 90 was being standardised parallel computer architecture was still evolving. Thus Fortran 90 standards committee consciously avoided including any parallel programming features in it.

However, after the publication of the Fortran 90 standard, a group called *High Performance Fortran Forum* (HPFF) was set up to extend Fortran to the emerging high performance parallel computers. An extension of Fortran 90 called *High Performance Fortran* (HPF) was proposed by HPFF. The main extensions were directives to Fortran 90 programs on parallelizing opportunities and some additional constructs. The Fortran standards committee decided on a strategy whereby a minor revision of Fortran 90 - Fortran 95 - will be prepared in the mid 90s and a major revision in the year 2001. The main revisions in Fortran 90 are to include some HPF features to enable writing parallelizable Fortran 90 programs. The other minor revisions are small corrections, clarifications and interpretations of some

Box 4 Upward Compatibility of Fortran

Whenever a new standard of a widely used programming language such as Fortran is published, programs written using the earlier version of that language are required to execute without change. This is necessary to protect investments made by the software industry to develop large application programs. Thus Fortran 90 standard stipulates that all Fortran 77 programs should execute without any change when Fortran 90 compilers are used. The standards committee, however, listed a number of statement types in Fortran as 'deprecated' and advised that these will not be accepted by Fortran compilers in the future. Accordingly, Fortran 95 has removed some of these as obsolete.

Fortran 95 has officially removed some of the features of Fortran 77 as non standard and not acceptable by its compilers. A draft of Fortran 95 standard was published in November 95 and the ISO standards documents in October 96.

of the statements made in the Fortran 90 standards document. All Fortran 90 programs will run without change when a Fortran 95 compiler is used.

Fortran 95 has officially removed some of the features of Fortran 77 as non standard and not acceptable by its compilers. A draft of Fortran 95 standard was published in November 95 and the ISO standards documents in October 96. Compilers for Fortran 95 are not yet widely available in the market. In what follows we will describe the important new features introduced in Fortran 95 which are extentions of Fortran 90.

FORALL Statement

Fortran 95 has included a statement which generalises the array assignment facility provided by Fortran 90. The statement looks like a DO loop but is quite different in the way it computes. It does not iterate on array indices sequentially like a DO loop. It computes the array expression on the right hand side of an assignment for all the specified indices and then performs the assignment. We give in Example 1 a FORALL statement which replaces the four interior elements of a (3×3) matrix by the sum of its neighbours.

Example 1

```
FORALL ( i = 2:3, j = 2:3 )
      x(i, j) = x(i, j –1 ) + x(i, j + 1 ) + x( i - 1, j )+x( i + 1,j)
END FORALL
```

The above statement calculates the right hand expressions of the above assignment as shown below:

$$x(2,1) + x(2,3) + x(1,2) + x(3,2) \quad (1)$$
$$x(2,2) + x(2,4) + x(1,3) + x(3,3) \quad (2)$$
$$x(3,1) + x(3,3) + x(2,2) + x(4,2) \quad (3)$$

$$x(3,2) + x(3,4) + x(2,3) + x(4,3) \qquad (4)$$

and assigns the value of expression (1) to x (2, 2), expression (2) to x (2, 3), expression (3) to x (3, 2) and expression (4) to x (3, 3). This is shown below using numerical values:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Matrix x before executing FORALL statement

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 24 & 28 & 8 \\ 9 & 40 & 44 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Matrix x after executing FORALL statement

The FORALL statement computes the array expression on the right hand side of an assignment for all the specified indices and then performs the assignment.

Observe that expressions (1), (2), (3), (4) can be calculated in any order as no sequencing is implied by the FORALL statement. Within the expression also individual terms can be evaluated in any order. Thus expressions (1), (2), (3), (4) may be computed in parallel in four processors of a parallel computer. FORALL statements may also be nested as shown in Example 2.

Example 2

```
FORALL ( i = 1 : 4 )
      x ( i, i ) = ABS ( x ( i, i ))
  FORALL ( j = i - 2 : i + 2, j /= i .AND. j > = 1 .AND. j <=4)
      x ( i, j) = x (i, i) * x ( j, j )
  END FORALL
END FORALL
```

The outer FORALL assigns values to x (1, 1), x (2, 2), x (3, 3) and x (4, 4) as per the statement:

$$x(i, i) = ABS(x(i, i))$$

This is shown with matrix x on top of the next page

A FUNCTION is said to have a side effect if, besides returning a value of the function, it alters the value(s) of some variable(s) which can be accessed outside the function.

$$\begin{bmatrix} -1 & 0 & 2 & -3 \\ 2 & -3 & -4 & 5 \\ -6 & 7 & -8 & -9 \\ 0 & -5 & 6 & 5 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 2 & -3 \\ 2 & 3 & -4 & 5 \\ -6 & 7 & 8 & -9 \\ 0 & -5 & 6 & 5 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 2 & -3 \\ 2 & 3 & -4 & 5 \\ 8 & 24 & 8 & 40 \\ 0 & 15 & 40 & 5 \end{bmatrix}$$

x at start | x after outer FORALL | x after inner FORALL

For each value of i, the inner FORALL computes indices j and the resultant values of (i, j) are: (i, j) = (3, 1), (3, 2), (3, 4), (4, 2) and (4, 3).

The matrix x after carrying out the statement x (i, j) = x (i, i) * x (j, j) is shown as the third matrix above.

Pure Functions

A FUNCTION is said to have a side effect if, besides returning a value of the function, it alters the value(s) of some variable(s) which can be accessed outside the function. If a FUNCTION has side effects, the order of execution of a FUNCTION is significant. Thus FUNCTIONs with side effects impede parallelization of a Fortran program, particularly when they are used in a FORALL statement. Thus Fortran 95 encourages development of side effect free FUNCTIONs and lets the user declare it as PURE, for example:

PURE FUNCTION f (x, y).

When a FUNCTION is declared PURE it is an assertion that it does not change any input arguments or global variables, has no I/O and has no SAVED variables. The Fortran 95 compiler is expected to check if any PURE FUNCTION has potential side effects and if so give an error message. All intrinsic functions (i.e. built in functions) in Fortran 90/95 are PURE.

A SUBROUTINE can also be declared PURE. A PURE SUBROUTINE can alter only INTENT (OUT) and INTENT (INOUT) arguments. PURE SUBROUTINES are primarily used by PURE FUNCTIONs in their computation.

Elemental Procedures

An elemental procedure is one that is specified using scalar dummy arguments but may also be called using array arguments. If the arguments are scalar then the result is also a scalar. If the argument(s) are arrays the shape of the result is the same as the shape of the argument(s). If there are two or more arguments they must be conformable. For array arguments, the function is applied to each element of the array and the result is an array. We give an example of an elemental subroutine in Example 3. This SUBROUTINE written using scalar dummy arguments, can be called using arrays as arguments and will interchange the two arrays. All elemental procedures are by definition pure. The main purpose of defining elemental procedures in Fortran 95 is to aid automatic parallelization of code by Fortran 95 compilers when the program is executed on parallel machines.

An elemental procedure is one that is specified using scalar dummy arguments but may also be called using array arguments.

Example 3

An elemental SUBROUTINE to interchange vectors

```
ELEMENTAL SUBROUTINE exchange ( x, y )
  REAL, INTENT (INOUT) :: x, y
  REAL :: temp
  temp = x
  x = y
  y = temp
END SUBROUTINE exchange
```

Many intrinsic functions in Fortran 90 are elemental, for example, trigonometric functions and exponential functions.

The other new features of Fortran 95 are minor and will not be discussed. Fortran 95 has declared certain Fortran 77 features as obsolete (see *Box 5*) and no new Fortran program should use these.

Box 5 Statements of Fortran 77 Declared as Obsolete in Fortran 95

- Fortran 77 used a fixed format for source statements. The format requires statements to be between column 7 and 72, comment character C in column 1 etc. This fixed format is obsolete. Fortran 95 statements use a free format and may be placed anywhere on a line.
- Computed GO TO statement available in Fortran 77 is not to be used. It has been replaced by the SELECT CASE statement.
- CHARACTER * form to specify length of strings in CHARACTER specification is obsolete. One should use instead CHARACTER (LEN = n) form.
- DATA statement used in Fortran 77 should not be used within executable statements.
- Arithmetic Statement Function available in Fortran 77 is obsolete.
- Results of character functions cannot have assumed length.
- The DO loop indices cannot be reals.
- Branching to an ENDIF statement from outside an IF block is not allowed.
- PAUSE statement of Fortran 77 is not allowed.
- ASSIGN statement and ASSIGNED GO TO statement are removed from the standard.
- H edit description in FORMAT statement is removed.
- ASSIGNED FORMAT is removed.

Suggested Reading

- **Metcalf M and Reid J.** ***Fortran 90/95 Explained.*** **Oxford University Press. Oxford, 1996.**
- **Rajaraman V.** ***Computer Programming in Fortran 90 and 95.*** **Prentice Hall of India. New Delhi, 1997.**

Address for Correspondence
V Rajaraman
Supercomputer Education & Research Centre
Indian Institute of Science
Bangalore 560 012, India
email:
rajaram@serc.iisc.ernet.in

Conclusions

Fortran is now the most popular high level language for numerical computations in science and engineering. Fortran 90/95 is a revolutionary change in Fortran. In the next standard which will probably be available in 2001 the complete upward compatibility assured for Fortran 77 programs by Fortran 90 will not exist. Future Fortran compilers will not accept Fortran 77 programs but Fortran 90/95 programs will be accepted without change.

Several generations of scientists and engineers have been using Fortran. In the process of evolution Fortran 90/95 designers have kept in view the special needs in numerical computing such as manipulation of multidimensional arrays and introduced language features to easily express algorithms which manipulate arrays. They have also introduced language features to allow compilers to generate optimized code for superscalar and parallel computing systems. Thus Fortran 95 and its successors will continue to be widely used in numerical computing.