Fortran History and Development

Ian Chivers and Jane Sleightholme - September 1st, 1995
Last update: September 2008

Development and Evolution of Fortran

Fortran was one of the first high level languages developed and widely adopted by the academic and scientific community. It was developed over a three year period (1954-1957) by a team at IBM lead by John Backus. Fortran stands for FORmula TRANslation and was used mainly by people with a scientific background for solving problems with a significant arithmetic content.

By 1966 and the first standard it was widely used, easy to teach, had demonstrated the benefits of subroutines and independent compilation, was relatively machine independent and often had very efficient implementations.

By the 1970s it had started to show its age in relation to some of the other languages that had emerged and there was pressure on the X3J3 committee to incorporate changes to bring it more into line with mainstream language development. It was standardised in 1978 (even though the next version was called Fortran 77) and whilst the changes that were made were on the welcome side many felt that they had not gone far enough.

Other languages emerged and established themselves. Pascal, Ada, Modula 2, C, C++ all became rivals to Fortran in the scientific and academic communities.

Fortran was next standardised in 1991 (yet again out by one) and called Fortran 90. This was major improvement and the changes are given below.

Work continued and 1996 saw the publication of the 1995 standard. Details are given below.

There are also two ISO Technical Reports that have been published on IEEE Floating Point Arithmetic and Allocatable Attributes. Details of these are given later.

Work on Fortran 2003 is complete and information is provided later.

Work on Fortran 2008 is in progress and details are given below.

The new features of Fortran 90

The new language features include:-

free source form:

names can be up to 31 characters in length

blanks are significant

lines up to 132 characters in length

up to 39 continuation lines

; as statement separator for multiple statements per line

! as comment symbol

include option for source text from files

modern control structures

Fortran 90 has a modern DO statement, with CYCLE and EXIT options and the control part of the DO can be conventional iteration, WHILE or no control clause. There is also a CASE statement.

specification of numeric precision

There is now a clean way to control numeric precision.

whole array processing

It is now possible to treat arrays as whole objects and simply write

```
A=B*SIN(A)
```

where A and B are arrays.

dynamic behaviour, including allocate, deallocate, pointers, recursion

user defined data types

modules, and with them come

operator overloading

generic procedures

The new features of Fortran 95

The last revision was finalised in 1996. It was a relatively small change compared to the changes between the Fortran 77 and Fortran 90 standard. There are some major features and some minor corrections and clarifications. Some of these were to keep Fortran in step with the work in the HPF area.

The major features include

FORALL statement and construct

pure and elemental user defined subprograms

initial association status for pointers

implicit initialisation of dervived type objects

Minor features include

new intrinsic function NULL

new intrinsic function CPU_TIME

automatic deallocation of allocatable arrays

SIGN can distinguish between +0 and -0

comments in namelist input data

references to pure functions in specification expressions

changes to some intrinsic functions

Details of the work in the areas of IEE Floating Point Arithmetic and Alloctable Attributes can be found at NAG.

http://www.nag.co.uk/nagware/NP/doc/TR.asp

These reports are now in the Fortran 2003 standard document.

The new features of Fortran 2003

The following is a verbatum extract from the September 2003 standard.

Fortran 2000 contains several extensions to Fortran 95; among them are:

> Derived-type enhancements: parameterized derived types (allows the kind, length, or shape of a derived type's components to be chosen when the derived type is used), mixed component accessibility (allows different components to have different accessibility), public entities of private type, improved structure constructors, and finalizers.
>
> Object oriented programming support: enhanced data abstraction (allows one type to extend the definition of another type), polymorphism (allows the type of a variable to vary at runtime), dynamic type allocation, SELECT TYPE construct (allows a choice of execution flow depending upon the type a polymorphic object currently has), and type-bound procedures.
>
> The ASSOCIATE construct (allows a complex expression or object to be denoted by a simple symbol).
>
> Data manipulation enhancements: allocatable components, deferred type parameters, VOLATILE attribute, explicit type specification in array constructors, INTENT specification of pointer arguments, specified lower bounds of pointer assignment and pointer rank remapping, extended initialization expressions, MAX and MIN intrinsics for character type, and enhanced complex constants.
>
> Input and output (i/o) enhancements: asynchronous transfer operations (allows a program to continue to process data while an i/o transfer occurs), stream access (allows access to a file without reference to any record structure), user specified transfer operations for derived types, user specified control of rounding during format conversions, the FLUSH statement, named constants for preconnected units, regularization of i/o keywords, and access to i/o error messages.
>
> Procedure pointers.
>
> Scoping enhancements: the ability to rename defined operators (supports greater data abstraction) and control of host association into interface bodies.
>
> Support for IEC 60559 (IEEE 754) exceptions and arithmetic (to the extent a processor's arithmetic supports the IEC standard).
>
> Interoperability with the C programming language (allows portable access to many libraries and the low-level facilities provided by C and allows the portable use of Fortran libraries by programs written in C).
>
> Support for international usage: (ISO 10646) and choice of decimal or comma in numeric formatted input/output.
>
> Enhanced integration with the host operating system: access to command line arguments and environment variables, and access to the processor's error messages (improves the ability to handle exceptional conditions).

A very readable article by John Reid, the Working Group 5 convenor can be found at

http://www.fortranplus.co.uk/resources/john_reid_new_2003.pdf

We've updated the original document so that the web links in this document are active, i.e. can be clicked on.

The New Features of Fortran 2008

Major features

Coarrays in the next standard:
ftp://ftp.nag.co.uk/sc22wg5/N1651-N1700/N1697.pdf

Rationale for coarrays in the next standard:
ftp://ftp.nag.co.uk/sc22wg5/N1701-N1750/N1702.pdf

Also in Fortran Forum, December 2007.

Other Features

Fortran 2008 contains several extensions to Fortran 2003; some of these are listed below.

The maximum rank of an array has been increased from seven to fifteen.

Performance enhancements: The DO CONCURRENT construct, which allows loop iterations to be executed in any order or potentially concurrently.

Pointers can be initialized to point to a target.

Performance enhancements: CONTIGUOUS attribute.

The intrinsic function ATAN is extended so that ATAN (Y, X) is ATAN2 (Y,X).

Allocatable components of recursive type.

The MOLD= specier has been added to the ALLOCATE statement.

OPEN statement enhancements that allow the processor to select a unit number when opening an external unit. Such a unit number is guaranteed not to interfere with any program-managed unit numbers.

The BLOCK construct (allows declarations within executable statements).

A disassociated or deallocated actual argument can correspond to an optional nonpointer nonallocatable dummy argument.

The concept of variable now includes references to pointer functions which return associated pointers.

The COMPILER VERSION and COMPILER OPTIONS functions provide information about the translation phase of the execution of a program.

The real and imaginary parts of a COMPLEX variable can be selected using a component-like syntax.

Scoped macros which can generate whole Fortran statements and subprograms.

The intrinsic function FINDLOC was added and a BACK= argument was also added to the intrinsic functions MAXLOC and MINLOC.

Parallel programming support: SPMD parallel programming, co-arrays for data exchange between images, image control statements, and collective procedures.

A BITS data type for non-numeric programming and enhanced handling of binary, octal, and hexadecimal constants.

The G0 edit descriptor.

Additional mathematical intrinsic functions for computing Bessel functions, error functions, the Gamma function, and generalized L2 norms.

Should I move to Fortran 90 and beyond?

Yes! In the same way that Fortran 77 displaced Fortran 66, Fortran 95 is displacing Fortran 77. How many people are still using Fortran 66 compilers?

As standard FORTRAN 77 is a complete subset it is possible to simply recompile your old code with the new compiler and develop new code in F90.

This protects your investment in your old, working Fortran 77 code.

The new features of the language bring it up to date and increase the range of problems that can be solved easily. The language provides a very good framework for modern software development. The separation of the implementation from the design is a very powerful tools for reliable software development. We see in the language the first steps to object orientation with encapsulation and a limited support for polymorphism, without the steep learning curve demanded by C++.