Fortran 1957 – 2008: A Language with a Past, Present and Future

Peter Crouch
pccrouch@bcs.org.uk
Chairman Fortran Specialist Group
www.fortran.bcs.org

BCS Birmingham Branch meeting 19 May 2008
<table>
<thead>
<tr>
<th>Year Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>Joined the British Computer Society</td>
</tr>
<tr>
<td>1997 - 2002</td>
<td>Chairman of the BCS Birmingham Branch</td>
</tr>
<tr>
<td>2002 - 2008</td>
<td>Chairman of the BCS Fortran Specialist Group</td>
</tr>
</tbody>
</table>
In the Beginning
Fortran Pioneers
IBM Films
Early Developments with example code
Standardisation
Modern Developments with example code
Applications
BCS Fortran Specialist Group
In the beginning the only practical way to program computers was in machine code, which was extremely tedious. The source code used octal notation.

By the 1950s assembly code had been developed, which was less tedious but still error prone and required a detailed knowledge of the computer hardware.
In late 1953, John Backus sent a brief letter to his boss at IBM, asking that he be allowed to search for a "better way" of programming computers, with a project timescale of six months. He got the nod and began the research project that would eventually produce FORTRAN.

As John Backus says in the film, “project completion was always six months away”!
Fortran Pioneers
John Backus' team in the late 1950s
FORTRAN, the first high level programming language, was announced to the computing world by John Backus and his team from IBM at the Western Joint Computer Conference held in Los Angeles, California in February 1957.

In mid-April 1957 the first documented delivery of the FORTRAN compiler for the IBM 704 took place to Westinghouse-Bettis for use in nuclear reactor design.
An IBM 704 mainframe
(image courtesy of LLNL)
An IBM 704 CPU from the 1950s
**Hollerith 80 column card**

<table>
<thead>
<tr>
<th>FORTRAN STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>6 7 8 9 10</td>
</tr>
<tr>
<td>11 12 13 14 15</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Creating the **IT** Profession
Frank Engel of Westinghouse, Pittsburg was concerned about the efficiency of the tape operations with the first FORTRAN compiler. He asked IBM if he could have a copy of the source code. They replied "IBM does not supply source code."

So Frank worked his way through an octal dump of the compiler and optimised the tape operations. The improvement so impressed IBM that they asked for a copy of the code, to which Frank replied "Westinghouse does not supply source code."
A copy of the 1982 IBM film in Windows Media Video format at 320 x 240 pixels resolution with a file size of 12.8 MB can be downloaded from the FORTRAN pages of the Computer History Museum website, www.softwarepreservation.org/projects/FORTRAN/video
Early Developments

1957    FORTRAN I

1958    FORTRAN II

1958    FORTRAN III - Not released to public

1961    FORTRAN IV - A "cleaned up" version of FORTRAN II

1962    First ASA FORTRAN standardization committee meets
Example code - FORTRAN I

```fortran
C     THE TPK ALGORITHM
C     FORTRAN I STYLE
FUNF(T)=SQRTF(ABSF(T))+5.0*T**3
DIMENSION A(11)
1 FORMAT(6F12.4)
READ 1,A
DO 10 J=1,11
   I=11-J
   Y=FUNF(A(I+1))
   IF(400.0-Y)4,8,8
4   PRINT 5,I
5   FORMAT(I10,10H TOO LARGE)
   GOTO 10
8   PRINT 9,I,Y
9   FORMAT(I10,F12.7)
10  CONTINUE
STOP 52525
```

Creating the IT Profession
```fortran
C       THE TPK ALGORITHM
C       FORTRAN IV STYLE
DIMENSION A(11)
FUN(T) = SQRT(ABS(T)) + 5.)*T**3
READ (5,1) A
1     FORMAT(5F10.2)
DO 10 J = 1, 11
    I = 11 - J
    Y = FUN(A(I+1))
    IF (400.0-Y) 4, 8, 8
4     WRITE (6,5) I
5     FORMAT(I10, 10H TOO LARGE)
    GO TO 10
8     WRITE(6,9) I, Y
     FORMAT(I10, F12.6)
10    CONTINUE
STOP
END
```
What FORTRAN 77 did for us

FORTRAN 77 added:

DO loops with a decreasing control variable (index)

Block IF statements - IF ... THEN ... ELSE ... ENDIF
   Before F77 there were only IF ... GOTO statements

Pre-testing of DO loops
   Before F77 DO loops were always executed at least once, so you had to add an IF ... GOTO before the loop if you wanted the expected behaviour

CHARACTER data type
   Before F77 characters were always stored inside INTEGER variables

Apostrophe delimited character string constants – 'Hello'

Main program termination without a STOP statement
PROGRAM TPK

C THE TPK ALGORITHM
C FORTRAN 77 STYLE
REAL A(0:10)
READ (5,*) A
DO 10 I = 10, 0, -1
   Y = FUN(A(I))
   IF (Y .LT. 400) THEN
      WRITE(6,9) I, Y
   ELSE
      WRITE (6,5) I
   ENDIF
10 CONTINUE
END
REAL FUNCTION FUN(T)
REAL T
FUN = SQRT(ABS(T)) + 5.0*T**3
END
<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1962</td>
<td>First ASA (later ANSI) standardization committee meets</td>
</tr>
<tr>
<td>1966</td>
<td>Publication of ANSI X3.9-1966 (FORTRAN 66) - first programming language standard</td>
</tr>
<tr>
<td>1978</td>
<td>Publication of ANSI X3.9-1978 (FORTRAN 77) – also published as ISO 1539:1980 – relatively minor revision</td>
</tr>
<tr>
<td>1991</td>
<td>ISO/IEC 1539:1991 (Fortran 90) - major revision</td>
</tr>
<tr>
<td>1997</td>
<td>ISO/IEC 1539-1:1997 (Fortran 95) - minor revision</td>
</tr>
<tr>
<td>2004</td>
<td>ISO/IEC 1539-1:2004 (Fortran 2003) - major revision</td>
</tr>
</tbody>
</table>
Modern Developments

Fortran 90 added:

- Free format source code form (column independent)
- Modern control structures (SELECT CASE & DO WHILE)
- Records/structures - called "Derived Data Types"
- Powerful array notation (array sections, array operators, etc.)
- Dynamic memory allocation
- Operator overloading
- Keyword argument passing
- The INTENT (IN, OUT, INOUT) procedure argument attribute
- Control of numeric precision and range
- Modules - packages containing data and code
PROGRAM TPK
!
The TPK Algorithm
!
Fortran 90 style

IMPLICIT NONE

INTEGER :: I

REAL :: Y

REAL, DIMENSION(0:10) :: A

READ (*,*) A

DO I = 10, 0, -1 ! Backwards
    Y = FUN(A(I))
    IF ( Y < 400.0 ) THEN
        WRITE(*,*) I, Y
    ELSE
        WRITE(*,*) I, ' Too large'
    END IF
END DO
CONTAINS ! Local function

FUNCTION FUN(T)
  REAL :: FUN
  REAL, INTENT(IN) :: T
  FUN = SQRT(ABS(T)) + 5.0*T**3
END FUNCTION FUN
END PROGRAM TPK
module Functions
public :: fun
contains

  function fun(t) result (r)
    real, intent(in) :: t
    real :: r
    r = sqrt(abs(t)) + 5.0*t**3
  end function fun

end module Functions

program TPK
!The TPK Algorithm
!F style
use Functions

integer :: i
real :: y
real, dimension(0:10) :: a
read *, a

do i = 10, 0, -1 ! Backwards
  y = fun(a(i))
  if ( y < 400.0 ) then
    print *, i, y
  else
    print *, i, " Too large"
  end if
end do

end program TPK
Fortran 2003 added:

- Support for object orientated programming
- Derived type enhancements
- Interoperability with C
- Data manipulation enhancements
- I/O enhancements including stream access
- Procedure pointers
- Support for IEEE 754 exceptions
- Support for international usage
- Enhanced integration with the host operating system including access to command line arguments
Fortran 2008 should include

- Coarrays as an extension for parallel processing
- Submodules to reduce compilation cascades
- Enhancements to aid optimisation
- Data enhancements including long integers, maximum array rank increased to 15, available kinds, hyperbolic and other functions
- I/O enhancements including getting unique unit numbers, new edit descriptors
- New BLOCK construct
- Bit manipulation procedures
- Execution of command line commands
Some application areas for Fortran

Weather forecasting and climate prediction
Analysis of seismic data for oil and gas exploration
Financial analysis
Vehicle crash simulation
Analysis of data from space probes
Modelling of nuclear weapons and test ban verification
Computational fluid dynamics, the “Numerical Wind Tunnel”
NEC SX-8 supercomputer as used by UK Met Office
BCS Fortran Specialist Group

The Group was founded in 1970 with the objectives of:

Forming a focus in the United Kingdom for work concerned with establishing and maintaining FORTRAN standards.

Working in association with national and international standardisation bodies.

The convenor (chairman) of the ISO WG5 committee responsible for the Fortran language is a member of the FSG committee as is the convenor of the BSI (UK) Fortran panel.

For the last few years the Fortran SG has provided financial support to enable several UK representatives to attend ISO meetings abroad.
In 2007 the Fortran SG was involved in a number of events and publications, as listed at www.fortran.bcs.org/2007/jubileeevents.php.

The largest of these was the 'Fifty Years of Fortran' meeting in January organised with the Computer Conservation Society. An audience of almost 60 heard 11 speakers talk about Fortran from the 1950s to the present day and into the future. The next two slides show some of the attendees and speakers.
If you want to know more

Modern open source and free Fortran compilers are available from a number of sources as are online tutorials.

The latest information on the next ISO Fortran standard is also available online.

Links to the above and more are available from the Resources page of the Fortran SG website at www.fortran.bcs.org/resources.php.
Acknowledgements

My grateful thanks go to Paul McJones of the Computer History Museum, Mountain View, CA, for providing me with the DVD version of the two IBM films.

Also I must thank my colleagues in the Fortran Specialist Group for their assistance and encouragement during my time as Chairman.