Fortran

Lexical/syntactical structures in Fortran and its history

Ondřej Fibich, Marek Olejník, 2012

History - early 50s

- low-level programming languages
  - dominance of assembly languages (AL)
  - problems with the backward compatibility
  - unsuitable for extensive programs
  - fast programs

- high-level programming languages
  - abstraction
  - easy to read, write, and maintain
  - lack of the performance

```
.model small.stack
100h
 .data
msg db 'Hello world!$'
 .codestart:
        mov    ah, 09h
        lea     dx, msg
        int     21h
        mov    ax, 4C00h
        int     21h

end start
```

"Hello world" example in DOS using MASM

History - creation

- abbreviation for "FORmula TRANslator"
  - scientific calculations and numerical applications
  - a high-level programming language
- compiler for FORTRAN released in 1954
  - the first optimizing compiler (performance, memory usage)
  - started a new computer science called *compiler theory*
- FORTRAN I released in 1957 (the first final release)
  - 5 times quicker for writing programs than assembly languages with only reduction 20% of the performance

```
program helloprint
*,"Hello World!"end
program hello
```

"Hello world" example in Fortran 77

History - the rise

- dominance among programming languages
  - advantages over assembly languages
  - derivatives – derivative language with user-specific functionality
  - re-issue with backward compatibility
- FORTRAN 66
  - the first standardized programming language (ANSI)
- evolution of the standard (FORTRAN 77, Fortran 90, etc.)
  - solved the backward compatibility isuues
  - new functionality is added according to needs of programmers (e.g. OOP)

History - today

- Fortran 2008
  - the standard and the language itself still evolves
  - modern programming language
- lost its dominance
  - the 22th favorite programming language
  - scientific programs (performance reasons)
- legacy for today
  - most current languages use principles implied from FORTRAN
  - compiler theory

Fortran structure

- Here is an example of Fortran code

```
program circle
real :: r, area
!This program reads a real number r and prints
!the area of a circle with radius r
read (*,*) r
area = 3.14159*r*r
write (*,*)' Area = ',area
stop
end program circle
```

  - Note that all variables are declared at the beginning of the program and before they are used

Declaration of Variables

- Single and double precision

```
real :: x
real(4) :: x
real*4 :: x
```

```
real (8) :: z
real*8 :: z
double precision :: z
```

- Complex number and characters

COMPLEX (COMPLEX*8 **or** COMPLEX(4))
COMPLEX(8) **or** COMPLEX*16
CHARACTER (LEN=n), CHARACTER (n), **or** CHARACTER*n
LOGICAL **can be** .TRUE. **or** .FALSE.

Declarating Arrays

- One-dimensional array

```
real :: a(4)

integer , parameter :: n=20
real :: a(n)

integer :: b(0:19)
```

- 2-dimensional array

```
double precision, dimension (10,10) :: c
```

Kind Parameter

- KIND parameter allow more flexibility for the user in declaring variable precision

```
integer, parameter :: i4=SELECTED_INT_KIND (4)
integer, parameter :: i8=SELECTED_INT_KIND (8)
integer, parameter :: r4=SELECTED_REAL_KIND(6,37)
integer, parameter :: r8=SELECTED_REAL_KIND(15,307)
integer (KIND=i4) :: ia
integer (KIND=i8) :: ib
real (KIND=r4) :: ra
real (KIND=r8) :: rb
print *,' Integer 4 ', huge (ia), kind (ia)
print *,' Integer 8 ', huge (ib), kind (ib)
print *,' Real 4 ', huge (ra), kind (ra)
print *,' Real 8 ', huge (rb), kind (rb)
```

Numeric Expressions

- Types of numeric expressions

+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponential

- Data type of Numeric Expressions
  - Combination of different data type

```
double precision :: x, y
y = x*2
```

  - The integer will be promoted to double

Loops and conditionals

- Do Loops

```
n = 10
do i=1, n
   …
enddo
```

```
do 5 i=1, n
   …
continue
```

- Do While statements

```
i = 0
do while (resid >= 5.0D-10)
    resid = abs (x(i))
    write (*,*) ' Continue execution '
    i = i+1
end do
```

Conditionals

- Logical expressions

.LT.	<	less than
.LE.	<=	less than or equal
.GT.	>	greater than
,GE,	>=	greater than or equal
.EQ.	==	equal
.NE.	/=	not equal

- Conditional (IF) Statements

```
if (resid < 5.0D – 10) stop
```

Functions and Subroutines

- Two types of subprograms in Fortran
  - Functions

```
real*8 :: x, y
x = func (y)
```

  - Subroutines

```
real*8 :: x, y
call subr (x, y)
```

Fortran I/O

- There are many ways of writing out data

```
print *,result
write (*,*) result
write (6,*) result
```

- Open files and read/write data

```
open (unit=2,file=‘ascii_data’,from=‘formatted’,status=‘old’)
read (2,’(10f20.4)’) (input (i), i=1,10)
close (2)

open (unit=3,file=‘binary_data’,form=‘unformatted’,status=‘unknown’,iostat=ierr)
if (ierr = 0) write (3,*) data
close (3)
```

Literature

- The FORTRAN Programming Language. University of Michigan [online]. 1999 [cit. 2012-12-03]. Available from: http://groups.engin.umd.umich.edu/CIS/course.des/cis400/fortran/fortran.html
- Fortran: A few historical details. *Numerical Algorithms Group* [online]. 2004 [cit. 2012-12-03]. Available from: http://www.nag.co.uk/nagware/np/doc/fhistory.asp
- *Programming Language Popularity* [online]. 2011 [cit. 2012-12-03]. Available from: http://langpop.com/
- Fortran Programming Language User Guide [online].

  [cit. 2012-12-04]. Available from:

  http://nf.nci.org.au/training/FortranBasic/slides/index.html