Fortran functions: some examples

Francesco Battista

Corso di Calcolo Numerico
[1]DIMA, “Sapienza” University of Rome, Italy

March 23, 2014

The first code: assegnazione.f90

```fortran
! File: assegnazione.f90
! Questo programma legge e stampa a schermo un numero
PROGRAM assegnazione

! Sezione dichiarativa
IMPLICIT NONE
INTEGER :: i

! Sezione esecutiva
WRITE(*,*) 'Scrivi un numero intero'
READ(*,*) i

WRITE(*,*) 'Hai scritto',i

! Sezione conclusiva
STOP
END PROGRAM assegnazione
```

The variables

- variable declaration by means **implicit none**
  - an **integer** variable is declared

- what is a variable?
  'It is a sort of paper on which you write with a pencil'

```
+--------+
| numero |     <—- Identifier or variable name
+--------+
|   -7   |     <—- variable value
+--------+
```

Assign a value to a variable

```
! File: assegnazione2.f90
! Questo programma legge due numeri intero e li stampa poi
!assegna il valore del primo moltiplicato per 10 al secondo
PROGRAM assegnazione2

! Sezione dichiarativa
IMPLICIT NONE
INTEGER :: num1, num2

! Sezione esecutiva
WRITE(*,*) 'Inserisci due interi (separati da spazio) e
    premi INVIO'
READ(*,*) num1, num2

WRITE(*,*) 'Hai scritto:',num1, num2

num2 = num1 * 10

WRITE(*,*) 'Le nuove variabili sono:',num1, num2

! Sezione conclusiva
STOP
END PROGRAM assegnazione2
```

assegnazione2.f90 code explanation

- Program structure:
  - the allocation section contains two variable of **integer** type
  - the execution section contains an allocation statement

- allocation structure
  variable = *statement*
  here the *statement* is a product

- How does the allocation operate?
  1. evaluate the statement
  2. assign the value of the statement to the varible in lef-hand-side:
     the variable is MODIFIED
     the statement is NOT MODIFIED

Substitute the statement `num2=num1*10` with the following ones:

ALPHA num2= num1+10

BETA num1= num1+10

GAMMA num2 + 10= num1

DELTA num2= num1 + num2

```
PRIMA     |                    DOPO
==========+==============================================
+-------+ | +-------+ | +-------+ | +-------+ | +-------+
| num1  | | | num1  | | | num1  | | | num1  | | | num1  |
+-------+ | +-------+ | +-------+ | +-------+ | +-------+
|   4   | | |       | | |       | | |       | | |       |
+-------+ | +-------+ | +-------+ | +-------+ | +-------+
          |           |           |           |
+-------+ | +-------+ | +-------+ | +-------+ | +-------+
| num2  | | | num2  | | | num2  | | | num2  | | | num2  |
+-------+ | +-------+ | +-------+ | +-------+ | +-------+
|  -7   | | |       | | |       | | |       | | |       |
+-------+ | +-------+ | +-------+ | +-------+ | +-------+
          |           |           |           |
          |   ALFA    |   BETA    |   GAMMA   |   DELTA
```

Variable types

The variable types are several, the most used are:

- **INTEGER**
- **REAL**
- **CHARACTER**
- **LOGICAL**: boolean: TRUE, FALSE
- the other are less used and are not useful for our aims

Fortran statement and examples

The statements are simple (atomic) or complex (not atomic)

Type	constant with name	constant without name	Variable	not atomic
Integer	dogs	3	i1	i1+i2+dogs
Real	pi	6.022E+23	r1	r1+r2*pi
Character	saluto	'a'	c1	saluto//c1//c2
Logical	vero	.TRUE.	l1	l1.AND.l2.OR.vero

```fortran
! Sezione dichiarativa
IMPLICIT NONE
INTEGER :: i1, i2
INTEGER, PARAMETER :: cats=44
REAL r1, r2
REAL, PARAMETER :: pi=3.1415
CHARACTER(10):: c1, c2
CHARACTER(10), PARAMETER :: saluto='ciao mondo'
LOGICAL :: l1, l2
LOGICAL,PARAMETER :: vero=.TRUE.
```

Integer type: details

- useful for countable quantities
- definition interval
  - maximum $2^{31} - 1 = 2'147'483'647$
  - minimum $-2^{31} = -2'147'483'647$
  - to represent an integer in single precision 4 bytes (32 bit) are used:
    * 2^{32} different values
    * half for positives and half fot negatives
    * among the positives one position is reverved to 0
- run-time possible problem: overflow

Real type: details

- suitable for physical quantities (temperature, pressure, density ...)
- definition interval
  - maximum about 10^{38}
  - minimum about -10^{38}
  - significant digits: about 7
  - to represent a real in single precision 4 bytes (32 bit) are used: 3 (24 bit) for the mantissa and 1 (8 bit) for the exponent
- several real format are admitted:
  - without exponent (default): 8314.23
  - with exponent: 8.31423E+3, where:
    * 8.31423 is the mantissa
    * 3 is the exponent
    * the base is 10
- the compiler is able to increase the bit number reserved to integer and real passing from 'single' to 'double' precision

Operations on reals and integers

- binary operations:
  + sum
  - subtraction
  * product
  / division
  ** exponential elevation
- unary operation: the sign plus(+) and minus(-)
- division between integer reads integer:
  7/3=2
- the operations occurs among constant with and without name and variables.
- the expression have to be linearized (writen on a row) and the product have to be written explicitly:

$$\frac{b^2 - 4ac}{h + 2a}$$

reads: `(b**2-4*a*c)/(h+2*a)`

Expressions evaluation rules: cerchio.f90

```
! File: cerchio.f90
!Questo programma legge un reale dallo schermo
!e calcola l'area e la circonferenza del cerchio
!di cui il reale e' il raggio
PROGRAM cerchio

! Sezione dichiarativa
IMPLICIT NONE
REAL :: radius
REAL, PARAMETER :: pi=3.141592

! Sezione esecutiva
WRITE(*,*) 'Qual Ã¨ il raggio del cerchio?'
READ(*,*) radius

WRITE(*,*) 'Il perimetro del cerchio e'':',2. * pi * radius
WRITE(*,*) 'L''area del cerchio e'':', pi * radius**2

! Sezione conclusiva
STOP
END PROGRAM cerchio
```

Expressions evaluation rules: cerchio.f

- How does the expression `pi*radius**2` read?
  1. `pi*(radius)**2`
  2. `(pi*radius)**2`

- priority among the operations:
  - '*' and '/' come before '+' and '-' hence:
    `6+4*2=6+(4*2)=14`
  - '**' comes before the others hence:
    `2*3**2=2*(3**2)=18`

- for same priority operations we have:
  - from left ot right for '+', '-' and '*', '/' hence:
    `6+4-2=8` or `6/2*3=9`
  - from right to left fot '**' hence:
    `3**2**3=3**(2**3)=6561`

Type conversion

- operation among different types (real/integer)
- implicit conversion
  `WRITE(*,*) 7.0*2` print 14.0 —2 is converted in 2.
  `WRITE(*,*) 1+1/2` print 1 —no conversion
  `WRITE(*,*) 1.+1/2` print 1.0 —0 is converted in 0.
  `WRITE(*,*) 1+1./2` print 1.5 —2 and 1 is converted in 2. and 1., respectively
- explicit conversion: proper conversion functions exist

Name	Domain	Codomain	Obtained Value
REAL(A)	INTEGER	REAL	A corresponding real
INT(A)	REAL	INTEGER	integer previous to A (truncation)
NINT(A)	REAL	INTEGER	integer closer to A (rounding)

Characters and strings

- allocation of a character of n length:
  `CHARACTER(n)nome_file`

- string operations

  * select a substring:
  `WRITE(*,*)nome_file(1:8)` print characters between from 1 to 8
  * interlock two string:
  `nome_file='ciao'//'.f'`

- from integer to character and return: the ASCII code

Name	Domain	Codomain	Obtained value
IACHAR(A)	CHARACTER(1)	INTEGER	ASCII code of A
ACHAR(A)	INTEGER	CHARACTER(1)	character which ASCII code is A

Main intrinsic function

Nome	Dominio	Codominio	Valore restituito	Note
COS(A)	R	R	cos(A)	A in radianti
SIN(A)	R	R	sin(A)	A in radianti
TAN(A)	R	R	tan(A)	A in radianti
ACOS(A)	R	R	arccos(A)	A in radianti
ASIN(A)	R	R	arcsin(A)	A in radianti
ATAN(A)	R	R	arctan(A)	A in radianti
EXP(A)	R	R	e^{A}	
LOG(A)	R	R	$\log_e A$	
LOG10(A)	R	R	$\log_{10} A$	
SQRT(A)	R	R	$\sqrt{A}$	
ABS(A)	R, I	R, I	$\|A\|$	
MOD(A,B)	I	I	resto di A/B	

...

and so on!!!

Conditional Statement

- PROBLEM 1 write a program which:
  * reads two integers
  * prints the maximum between them
- EXEMPLA:
  * reads $-3, 2$ print 2
  * reads $2, -3$ print 2
  * reads $5, 5$ print 5

- PROBLEM 2: write a program which:
  * reads an integer
  * prints the message 'even' or 'odd'

massimo.f90 code: IF-THEN-ELSE

```
! File: massimo.f90
! Calcolo del massimo dati due numeri
PROGRAM massimo

! Sezione dichiarativa
IMPLICIT NONE
INTEGER :: num1, num2, massimo

! Sezione esecutiva
WRITE(*,*) 'Inserisci due interi (separati da spazio)'
READ(*,*) num1, num2

IF (num1.GT.num2) then
  massimo = num1
ELSE
  massimo = num2
ENDIF

WRITE(*,*) 'Il massimo e'':',massimo

! Sezione conclusiva
STOP
END PROGRAM massimo
```

max.f code: IF-THEN-ELSE

- The code is clearer with indentation than without.
- Sintassi:

```
IF (logical expression) THEN
        statements
ELSE
        statements
ENDIF
```

- Observations
    - THEN is on the same raw of IF
    - ENDIF is demanding
    - ELSE is not demanding
- The type of the 'logical expression' is LOGICAL (.true. or .false.)

Logical expressions

exempla of logical expression

- 5 .gt. 2 —> .TRUE.
- 5 .lt. 2 —> .FALSE.
- 5 .eq. 2+3 —> .TRUE.
- 5 .gt. 2.0 —> .TRUE.

Logical operations (six):

.eq. uguale

.lt. minore

.gt. maggiore

.ne. diverso

.le. minore o uguale

.ge. maggiore o uguale

1. The left and right terms have to be of the same type REAL:REAL CHARACTER:CHARACTER
2. The automatic conversion rules apply also in this case REAL:INTEGER is allowed

TAKE CARE THE LIMITED PRECISION IN THE REAL REPRESENTATION

Complex logical conditions

- Logical 'IF' sintax:
  IF (logical expression) istruction
  - only one row
  - only one istruction
  - nor ELSE and END IF
- it is possible to combine the logical expression: relational operators
  - .AND. .OR.: binary
  - .NOT.: unary, it has the priority in .AND., .OR.

.AND.	.FALSE.	.TRUE.
.FALSE.	.FALSE.	.FALSE.
.TRUE.	.FALSE.	.TRUE.

.OR.	.FALSE.	.TRUE.
.FALSE.	.FALSE.	.TRUE.
.TRUE.	.TRUE.	.TRUE.

.NOT.	.FALSE.	.TRUE.
	.TRUE.	.FALSE.

- the logical expressions have the priority on the relational operators