Fortran functions: some examples

Francesco Battista

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

March 23, 2014

Cicle statements

- two type of control statements:
  - conditional istructions (IF and CASE not discussed here)
  - cicle istructions (definite and indefinite iteration)
- cicle istructions are useful for iterative istructions
- read three number and print the sum
  - use of known istructions
  - use the cicle istructions **somma.f90**

IF-THEN-ELSE statement: `even_odd.f90` code

```
1  ! file even_odd.f
2  ! This code reads one integer and say if it is even or odd
3  PROGRAM even_odd
4  !sezione dichiarativa
5  IMPLICIT NONE
6  INTEGER i1
7  CHARACTER(4) num
8  !sezione esecutiva
9  WRITE(*,*) 'Insert one integer, then press ENTER'
10 READ(*,*) i1
11 if (mod(i1,2).eq.0) then
12    num='even'
13 else
14    num='odd'
15 endif
16 write(*,*) 'the inserted number is ',num
17 !sezione conclusiva
18 STOP
19 END
```

Cicle statemets: `somma.f90` code

```
1  !file: somma.f
2  !Show the cicle statements use
3   PROGRAM somma
4  !Declaration section
5   IMPLICIT NONE
6   INTEGER amount
7   INTEGER num1
8   INTEGER summ, i
9  !Execution section
10  amount=3
11  WRITE(*,*)'Insert', amount ,'integer and press each time ENTER:
        '
12  summ=0
13  DO i=1,amount
14  READ(*,*) num1
15  summ = summ + num1
16  ENDDO
17  WRITE(*,*) 'The sum reads:', summ
18 !End section
19  STOP
20  END
```

Definite cicle: introduction

- a variable is necessary, namely the **counter**: it is the cicle index
- cicle index:
  1. automatic initialization
  2. automatic updating each cicle
  3. the user does not never change it
- start and end of the cicle:
  1. can be chosen in run-time
  2. cannot be changed once chosen

Definite cicle: syntax

example:

DO index = start, end [, increase]
statement sequence
ENDDO

- the index is of **INTEGER** type
- start, end, increase are expressions of integer type
- increase is optional, it is 1 by default
- none among start, end, increase can be changed in *s*tatement sequence

Definite cicle: semantic

- **Simple Case**: increase=1 (The increase can assume also negative value)
- What does implicitly occur in the DO-LOOP?
  - before DO-LOOP
    index = start
    This occurs only the first time
  - IF index $\leq$ end THEN
    statement sequence
    ELSE end of the cicle.
    This occurs every iteration of the DO-LOOP
  - before the ENDDO
    index = index+1
    This occurs every iteration of the DO-LOOP

Undefinite cicle

- **Definite cicle**: the number of iteration is known

- some operations are impossible with this istruction

  FOR EXAMPLE: read on scream a sequence of number ending with 0 end count the length of the sequence.

- **Undefinite cicle**: during the esecution the number of iteration is not necessarly known

Undefinite cicle: syntax

example:

DO
    statement sequence
IF (logical expression) EXIT
    statement sequence
ENDDO

- the statement sequeces are not necessary
- the locical expression "sentinel": represent a peculiar event, e.g. a particular value of a variable
- there are other ways to do this

Definite DO-LOOP: fattoriale.f90 code

```
1  !file: fattoriale.f
2  !This program reads a number and compute the factorial
3  PROGRAM fattoriale
4  !sezione dichiarativa
5  IMPLICIT NONE
6  INTEGER:: i1, l
7  INTEGER:: fact
8  !sezione esecutiva
9  WRITE(*,*) 'Write the number on the screen and press ENTER'
10 READ(*,*) i1
11 fact=i1
12 DO l=1,i1-1
13    fact=fact*(i1-l)
14 ENDDO
15 if (i1.eq.0) fact=1
16 WRITE(*,*) 'The factorial of', i1,'is:',fact
17 !sezione esecutiva
18 STOP
19 END PROGRAM fattoriale
```

Undefinite DO-LOOP: sequenza.f90 code

```
1  ! file: sequenza.f
2  ! This program is an example of the undefined DO-LOOP
3  PROGRAM sequenza
4  !declaration section
5  IMPLICIT NONE
6  INTEGER bit !read element sequence
7  INTEGER cont !0 sequence length
8  INTEGER maxleng
9  cont = 0
10 maxleng = 0
11 !execution section
12 WRITE(*,*) 'Insert a 0,1,2 sequence (press ENTER each time)'
13 DO
14   WRITE(*,*) 'Insert a number (no 0 or 1 to end):'
15   READ(*,*) bit
16   IF (bit .ne. 1 .and. bit .ne. 0) EXIT
17   IF (bit .eq. 0) THEN
18     cont = cont + 1
19     IF (cont .gt. maxleng) maxleng = maxleng + 1
20   ELSE
21     cont = 0
22   ENDIF
23 ENDDO
24 WRITE(*,*) 'The most length sequence of only 0 counts'&
25             ,maxleng,' zeros'
```

Problems: definite and undefinite DO-LOOP

- **Problem 1** definite DO-LOOP:
  print the firsts n (chosen by the user) even number
  1. read the number n from screen
  2. write the number 2*n in a DO-LOOP

- **Problem 2** undefinite DO-LOOP:
  read from screen two integers and print the greater common divisor
  1. read the number two integer a, b
  2. set the minimun between them equal m
  3. if mod(a,m) and mod(b,m) are equal 0 then exit
  4. if one of mod(a,m) or mod(b,m) is not equal 0 then m=m-1
  5. then repet from point 3

File and formatting

- these concepts are independent on the programmation language
- **file**: a sequence of elements (es. characters) saved on the secondary memory
- **text file**: each thinks can be read from keyboard and written on the screen (integer, real, characters ...)
- data access:
  1. sequential from the start to the end
  2. casual
- **physical name** restriction on the lenght of the file physical name (Unix is CASE SENSITIVE)
- **logical name** identifier internal to the program

Operations on files

- **Open a file**: make a link between the physical and logical name
  1. read only
  2. write only:
     * 'NEW' mode, the file does not exist, a new file is created
     * 'REPLACE' modem, the file is rewritten independly on the existence of the file, deleting the old content
  3. read/write mode
- read of file content (when it is allowed)
- write of file content (when it is allowed)
- file closing: remove the link

Open a file

- Istruction: OPEN
- some features should be specified:
  1. UNIT logical name (non negative INTEGER)
  2. FILE physical name
  3. STATUS
     - OLD: read only
     - NEW: write only, the file does not exists and is created
     - REPLACE: write only, the file is (re)written
     - SCRATCH: the file is not saved on disk

Other operations

- Closing: CLOSE(unit)
- writing; WRITE(unit,format)
- reading; READ(unit,format)
- go to the start of the file: REWIND(unit)
- **EXERCISE** read a file with a number sequence and print the sum of them

READ statement: `sommafile.f90` code

```
1  !file: sommafile.f
2  !read a sequence of a number from a file and print the sum
3  PROGRAM sommafile
4  !declaration section
5  IMPLICIT NONE
6  CHARACTER(12) file_name
7  PARAMETER(file_name='data.dat')
8  INTEGER dato, amount, summ
9  amount=0
10 summ=0
11 !execution section
12 WRITE(*,*) 'Read a file ', file_name
13 OPEN (UNIT=1,FILE=file_name,STATUS='old',ACTION='read')
14 DO
15 READ(1,*,END=99) dato
16 amount = amount + 1
17 summ   = summ   + dato
18 ENDDO
19 99    CONTINUE
20 CLOSE(1)
21 WRITE(*,*) 'The file is made by',amount,'data'
22 WRITE(*,*) 'Their sum is',summ
23 !end section
24 STOP
25 END
```

Free format print

```
INTEGER i
REAL pigreco
i=21
pigreco=3.14159
WRITE(*,*)' i vale =',i,'; pi greco vale =', pigreco
! 'i vale =',          21,'; pi greco vale =',          3.14159
```

- print in free format = '*', the second argument of WRITE(,)
- a lot of unuseful spaces
- the number of space is uncontrolled
- not suitable for ordered print

Formated print

```
      INTEGER i
      REAL pigreco
      i=21
      pigreco=3.14159
      WRITE(*,100) i, pigreco
100   FORMAT(' i vale =',I3,'; pi greco vale =',F7.3)
      ! 'i vale =', 21,'; pi greco vale =', 3.142
```

- **100** label of the FORMAT istruction
- **I3** descriptor of an integer format
  it uses 3 characters with right align
- **F7.3** descriptor of the real format with floating point
  it uses 7 characters with 3 after the point and right align
- rounding of real in writing not in memory

INTEGER descriptor I

- general form **rIw.m**
  r, w and m are integer constant without name

Iw it prints using w character sign included with right align

Iw.m it prints using w character sign included with at least m characters that are nor ' ' (space) neither sign

rIw it prints r times using w character sign included

Other descriptors

		NOTE	ALLIN.
REAL	rFw.d	Virgola fissa	D
	rEw.d	Virgola mobile, 0.1 <= |mantissa| < 1.0	D
	rESw.d	Virgola mobile, 1.0 <= |mantissa| < 10.0	D
LOGICAL	rLw	w >=1 (stampa 'T' o 'F')	D
CHARACTER	rAw	Stampa w caratteri	D
	rA	Stampa tutti i caratteri necessari	S

r number of repetitions

w number of used characters (included '+','-','.','E'

d number after the point