Fortran functions: some examples

Francesco Battista

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

March 23, 2014

ERRATA CORRIDGE: fattoriale.f90 code

```fortran
!file: fattoriale.f
!This program reads a number and compute the factorial
PROGRAM fattoriale
!sezione dichiarativa
IMPLICIT NONE
INTEGER:: i1, l
INTEGER:: fact
!sezione esecutiva
WRITE(*,*) 'Write the number on the screen and press ENTER'
READ(*,*) i1
fact=i1
DO l=1,i1-1
   fact=fact*(i1-l)
ENDDO
if (i1.eq.0) fact=1
WRITE(*,*) 'The factorial of', i1,'is:',fact
!sezione esecutiva
STOP
END PROGRAM fattoriale
```

Undefinite DO statement: `mcd.f90` code

```fortran
!file: mcd.f
!This program prints the greater common divisor of two integers
PROGRAM mcd
!sezione dichiatativa
IMPLICIT NONE
INTEGER a, b, m
!sezione esecutiva
WRITE(*,*) 'Insert two number separated by a space:'
READ(*,*) a, b

m=a
IF (b.lt.a) m=b
DO
IF (mod(b,m).eq.0.and.mod(a,m).eq.0) EXIT
m = m-1
ENDDO
WRITE(*,*) 'The greater common divisor is',m
!sezione conclusiva
STOP
END
```

Modules: subprograms

- modules are useful for engineering applications

- they are useful for public use

- the modules is a black block:
  - it is important WHAT it does
  - it is not important HOW

- the subprograms are useful in two cases:
  - to do more than once the same istructions
  - for parametric istructions

Subprogramming pro

- it is useful for semantic errors correction
- the subprogram can be re-used in other contexts
- two types of subprogram exist:
  - SUBROUTINE
  - FUNCTION
- can be written in a different file '.f90':
  in this case it have to be COMPILED and LINKED with the main program
- can be written in the same file of the main program:
  the compilation of the unique file containing all is required

Subprograms

- every subprogram is called in the main program (or in other subprograms)
- two variable are defined
  - dummy variable: useful for communications between the calling program and the subprogram, NO MEMORY IS USED
  - local variable: used in the subprograms and deallocated at the end of the subprogram, THE MEMORY IS TEMPORARILY OCCUPIED

```
! in the main program
...
CALL nome_subroutine(arguments)
...

!The subroutine program
SUBROUTINE nome_subroutine(arguments)
IMPLICIT NONE
...
...
RETURN
END
```

subroutine & fucntion

- arguments have to be declared both in the main program and in the subroutine
- no STOP at the end but RETURN
- the rules are the same of the main program

Some differences are in the FUNCTION case:

- to the function a value is associated
- the type of this value (i.e. of the function) have to be declared
- the function is called in the expression
- in the calling the key word CALL is not used
- the function returns only one value at time, in the other cases the subroutine subprogram should be used