

# PROGRAMMING FOR COMPUTATIONAL CHEMISTRY

Loops and choices

Emanuele Coccia

Dipartimento di Scienze Chimiche e Farmaceutiche

# Do loop

- Construct to **repeat** instructions within a loop  
    *do i=m,n,l*  
        *instructions*  
    *...*  
    *enddo*
- $i, m, n$  and  $l$  are integers
- $l$  is optional (default 1)

# Do loop

- Construct to **repeat** instructions within a loop  
    *do i=m,n,l*  
        *instructions*  
    *...*  
    *enddo*
- *i*, *m*, *n* and *l* are integers
- *l* is optional (default 1)
- **Counter** *i* controlling the numbers of iterations
- The counter can not be modified

# Do loop

- Construct to **repeat** instructions within a loop

```
do i=m,n,l
    instructions
    ...
enddo
```
- $i, m, n$  and  $l$  are integers
- $l$  is optional (default 1)
- **Counter**  $i$  controlling the numbers of iterations
- The counter can not be modified
- Nested loops are allowed

# Do loop

- Construct to **repeat** instructions within a loop  
    *do i=m,n,l*  
        *instructions*  
    *...*  
    *enddo*
- *i*, *m*, *n* and *l* are integers
- *l* is optional (default 1)
- **Counter i** controlling the numbers of iterations
- The counter can not be modified
- Nested loops are allowed
- **Exit** instruction: exit the loop

# Do loop

- Construct to **repeat** instructions within a loop  
    *do i=m,n,l*  
        *instructions*  
    ...  
    *enddo*
- *i*, *m*, *n* and *l* are integers
- *l* is optional (default 1)
- **Counter** *i* controlling the numbers of iterations
- The counter can not be modified
- Nested loops are allowed
- **Exit** instruction: exit the loop
- **Cycle** instruction: transfer control to **enddo**

# Do loop

- Construct to **repeat** instructions within a loop  
    *do i=m,n,l*  
        *instructions*  
    ...  
    *enddo*
- *i*, *m*, *n* and *l* are integers
- *l* is optional (default 1)
- **Counter** *i* controlling the numbers of iterations
- The counter can not be modified
- Nested loops are allowed
- **Exit** instruction: exit the loop
- **Cycle** instruction: transfer control to **enddo**
- Examples **do1.f90** and **do2.f90**

# Do while construct

- Loop with **exit condition**

*do while(condition)*

*instructions*

*...*

*enddo*

- Instructions repeated until the condition (logical expression) is **true**

# Do while construct

- Loop with **exit condition**

*do while(condition)*

*instructions*

*...*

*enddo*

- Instructions repeated until the condition (logical expression) is **true**
- Example **dowhile.f90**

# If condition

- Do instructions if a condition (logical expression) is **verified**  
*if (condition) then*  
    *instructions*  
    *...*  
*endif*

# If condition

- Do instructions if a condition (logical expression) is **verified**  
*if (condition) then*  
    *instructions*

...

*endif*

- **Multiple** options  
*if (condition 1) then*  
    *instructions 1*

...

*elseif (condition 2) then*  
    *instructions 2*

...

*else*  
    *instructions 3*

...

*endif*

# If condition

- Do instructions if a condition (logical expression) is **verified**  
*if (condition) then*  
    *instructions*

...

*endif*

- Multiple** options  
*if (condition 1) then*  
    *instructions 1*

...

*elseif (condition 2) then*  
    *instructions 2*

...

*else*  
    *instructions 3*

...

*endif*

- Also allowed: *if (condition) instruction*

# If condition

- Do instructions if a condition (logical expression) is **verified**  
*if (condition) then*  
    *instructions*

    ...  
*endif*

- **Multiple** options  
*if (condition 1) then*  
    *instructions 1*

    ...  
*elseif (condition 2) then*  
    *instructions 2*

    ...  
*else*  
    *instructions 3*

    ...  
*endif*

- Also allowed: *if (condition) instruction*
- Examples **if.f90**, **exit.f90**, **repeat.f90**, **cycle.f90** and **nested.f90**

- Multiple options

```
select case (expression)
  case (choice1)
    instructions 1
    ...
  case (choice2)
    instructions 2
    ...
end select
```

- Multiple options

```
select case (expression)
  case (choice 1)
    instructions 1

  ...

  case (choice 2)
    instructions 2

  ...

end select
```
- Example `case.f90`

- fattoriale.f90

# Examples

- fattoriale.f90
- fibonacci.f90

# Examples

- fattoriale.f90
- fibonacci.f90
- norma\_vettore.f90

# Examples

- fattoriale.f90
- fibonacci.f90
- norma\_vettore.f90
- prodotto\_scalare.f90

- fattoriale.f90
- fibonacci.f90
- norma\_vettore.f90
- prodotto\_scalare.f90
- matmul.f90