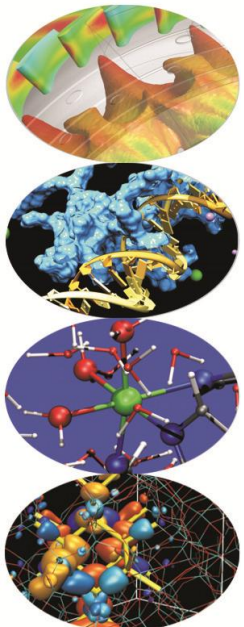


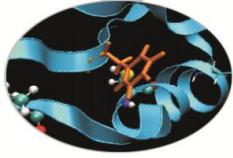
Le operazioni di Input e Output

Introduction to Fortran 90

Paolo Ramieri, *CINECA*

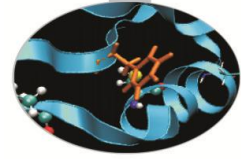
Marzo 2015





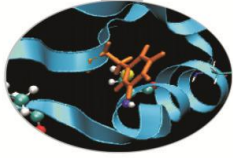
Input e Output

- Il **salvataggio dei dati** su disco e la loro lettura è possibile dopo aver generato un "collegamento" tra le aree disco ed il programma.
- Accanto alle unità di I/O corrispondenti ad aree disco esistono unità predefinite, che corrispondono a dispositivi di I/O particolari, quali stampanti, schermi, tastiere.



Input e Output

- Un programma legge il valore delle variabili dall'esterno (**INPUT**), le elabora, quindi scrive i risultati verso l'esterno (**OUTPUT**)
- Input e Output di norma leggono e scrivono da/verso file
- Il file deve essere preventivamente indicato e associato con una "file unit"
- Tutte le operazioni di I/O fanno riferimento alla "file unit"

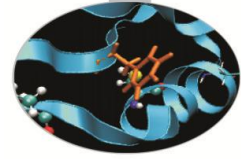


Input e Output

- OPEN (FILE =“mydata.in”,UNIT=3)
- READ (UNIT=3,*) date
- WRITE (UNIT=4,*) results
- CLOSE (UNIT=3)

- 5 → std-in
- 6 → std-out
- 0,3 → std-err

Non serve OPEN



L'istruzione READ

L'istruzione READ serve per la lettura di dati (**input**)

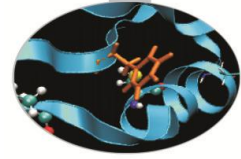
Sintassi:

READ (elenco clausole) {input}

dove le clausole principali sono:

Read (*,*) ncasi

- UNIT = numero (* = unit5 – std-in)
- FMT = formato (* = variable driven)
- IOSTAT = valore (opzionale)
- END = valore_label (opzionale)



L'istruzione READ

Le clausole devono essere riportate con il loro nome.

Esempio 1:

```
READ (UNIT=41, FMT=" (F10.2) ", END=99) anno
```

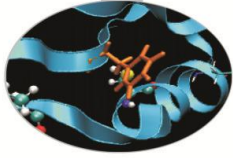
Possono fare eccezione UNIT e FMT purché siano, rispettivamente la prima e la seconda clausola.

Esempio 2:

```
READ (41, " (F10.2) ", END=99) anno
```

Esempio 3:

```
READ (41, " (3 (F10.2) ) ", END=99) aa, mm, gg
```



L'istruzione READ

La clausola `UNIT` rappresenta l'unità su cui si vuole operare per leggere o scrivere dati.

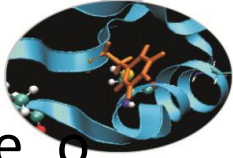
Spesso è associata ad un **file** a cui è necessario accedere con l'istruzione `OPEN`.

L'uso dell'**asterisco** indica la **periferica di default (stnd-in)**:

Esempio 1:

```
READ (*, "(F10.2)", END=99) anno
```

L'istruzione READ



La clausola `FMT` determina il **formato** dei dati da leggere o scrivere. Il formato può essere specificato **all'interno dell'istruzione di lettura/scrittura** (esempio 1) oppure richiamando l'istruzione `FORMAT` tramite un'**etichetta** (2).

Esempio 1:

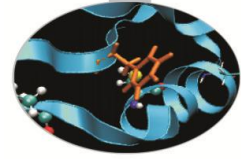
```
READ (41, " (F10.2) ", END=99) anno
```

Esempio 2:

```
READ (18, 60) anno  
60 FORMAT (8X, I4)
```

Esempio 3

```
CHARACTER (*), parameter :: fmt1="(f10.2)"  
...  
READ (*, FMT=fmt1) anno
```

L'istruzione READ

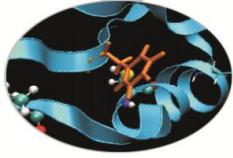
L'uso dell'**asterisco** come formato indica che i dati sono letti in **formato libero**.

Esempio:

```
READ (*, *) anno
```

La clausola **IOSTAT** ritorna un valore intero, indicativo dell'**esito della lettura** (**0** in caso di successo)

Si può specificare in **END** l'etichetta a cui passare il **controllo** dell'esecuzione nel caso in cui la lettura giunga al termine del file.



L'istruzione WRITE

L'istruzione WRITE serve per la scrittura di dati (**output**)

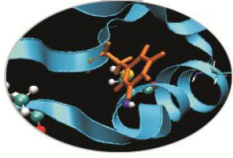
Sintassi:

```
WRITE (elenco clausole) {output}
```

dove le clausole principali sono:

- UNIT = numero (* = unit6 – std-out)
- FMT = formato (* = variable driven)
- IOSTAT = valore (opzionale)

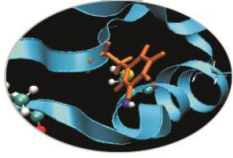
PRINT *, "Hello World"



L'istruzione WRITE

Le clausole hanno lo stesso significato e utilizzo delle clausole omonime dell'istruzione `READ`.

ERR specifica il valore di un'etichetta a cui passare il **controllo** dell'esecuzione nel caso in cui vi sia un errore in fase di scrittura.



L'istruzione WRITE

Esempio 1:

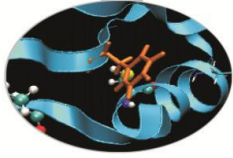
```
WRITE (*, *) anno
```

Esempio 2:

```
WRITE (41, "(F10.2)", ERR=99) anno
```

Esempio 3:

```
WRITE (18, 60) (anni(i), i=1, 10)  
60 FORMAT (8X, I4)
```



L'istruzione PRINT

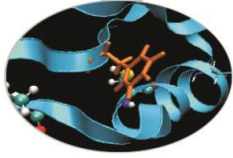
Solamente per la scrittura a **standard output** esiste anche l'istruzione `PRINT` con una sintassi molto più **semplice**:

```
PRINT format, {output}
```

Esempio:

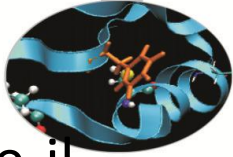
```
PRINT *, anno
```

Esercizi



1. Scrivere un programma che legga dati da tastiera e li scriva a video. Utilizzare il formato libero
2. Al programma precedente, aggiungere la stampa del valore di `IOSTAT`, sia per le istruzioni `READ`, sia per le istruzioni `WRITE`.

L'istruzione OPEN



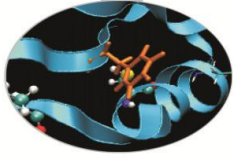
L'istruzione `OPEN` realizza un collegamento tra le aree disco e il programma, associando un determinato file ad un numero. Questo numero, chiamato unit (**unità di I/O**) da quel momento in poi individua quel file in tutte le operazioni di I/O (**fino al CLOSE**).

Sintassi:

`OPEN (elenco clausole)`

dove le clausole principali sono:

- `UNIT = numero (necessaria)`
- `FILE = nome (necessaria)`
- `STATUS = stringa`
- `ACTION = stringa`
- `IOSTAT = valore`



L'istruzione OPEN

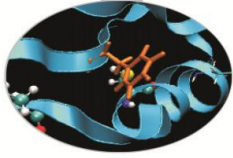
UNIT indica il numero di unità associato al file: sarà lo stesso numero da indicare come UNIT nelle istruzioni di READ o WRITE.

FILE indica il nome del file a cui si deve accedere (comprensivo di path, anche relativo). Lo stesso file non può essere connesso a 2 unità diverse.

Esempio:

```
OPEN (UNIT=100, FILE="input.dat")
```

```
WRITE (100, *) anno
```

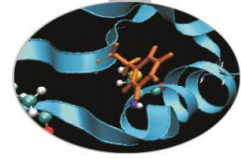
L'istruzione OPEN

IOSTAT restituisce alla variabile indicata un numero intero: quando è pari a **zero**, l'operazione richiesta ha avuto **successo**, quando invece è un valore positivo, allora l'operazione di apertura del file è fallita.

STATUS indica se il file da aprire deve esistere già oppure se deve essere creato o deve rimpiazzarne uno esistente.

I soli valori possibili sono:

OLD, REPLACE, NEW, SCRATCH, UNKNOWN



L'istruzione OPEN

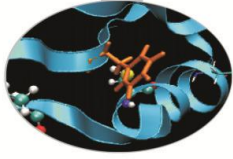
ACTION indica la modalità con cui operare su quel file, ovvero se si agirà in sola lettura, in sola scrittura o in entrambi i modi.

I soli valori possibili sono:

READ, WRITE, READWRITE

Esempio:

```
OPEN (UNIT=100, FILE="input.dat",  
      STATUS="OLD", ACTION="READ", IOSTAT=var)
```



L'istruzione CLOSE

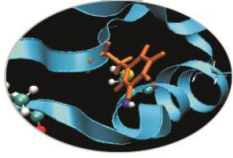
CLOSE chiude l'unità e libera il numero associato ad essa. Se tale istruzione non viene usata, al termine del programma l'unità viene chiusa automaticamente.

Sintassi:

`CLOSE (elenco clausole)`

dove le clausole principali sono:

- `UNIT = numero (necessaria)`
- `STATUS = stringa`



L'istruzione CLOSE

STATUS indica se mantenere o rimuovere il file dopo la sua chiusura.

I soli valori possibili sono:

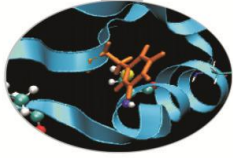
KEEP, DELETE

Esempio:

```
CLOSE (UNIT=100, STATUS="KEEP")
```

```
CLOSE (100, STATUS="KEEP")
```

Esercizi



3. Scrivere un programma che legga dati da tastiera e li scriva su un file. Utilizzare il formato libero
4. Scrivere un programma per la conversione di una qualsiasi temperatura da gradi Fahrenheit in gradi Kelvin e Celsius. Leggere il dato di temperatura da un file e stampare i risultati sul video.

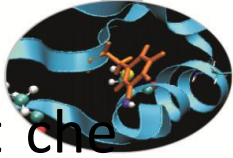
Conversione in gradi Kelvin:

$$\text{temp_k} = (5. / 9.) * (\text{temp_f} - 32.) + 273.15$$

Conversione in gradi Celsius:

$$\text{temp_c} = \text{temp_k} - 273.15$$

Formato



I valori delle variabili sono mantenute come stringhe di bit che vengono interpretate in modo diverso a seconda del tipo della variabile (Formato BINARIO).

Durante le operazioni di I/O spesso è necessario trasformare il formato BINARIO in un formato leggibile (fORMATTATO).

Questa conversione puo' avvenire in diversi modi e viene comandata tramite il campo "FORMAT"

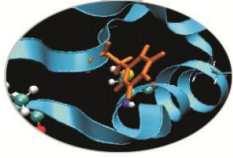
Il formato dei dati nelle istruzioni di input e output può essere specificato:

- con l'istruzione `FORMAT`, preceduta da una label

```
100 FORMAT (10x, i3)
```

- all'interno dell'istruzione `READ/WRITE`

```
FMT=" (I3, 2x, F10.2) "
```



Descrittori del formato

In entrambi i casi la **sequenza del formato** è costituita da una serie di **descrittori** dei dati e di funzioni di controllo che specificano quali sono i tipi di dati che verranno letti o scritti e in che modo dovranno essere disposti (per esempio, su quante righe, con quale intervallo...).

Una cifra posta prima del descrittore indica quante volte deve essere ripetuto il descrittore in oggetto (**fattore di ripetizione**).

Descrittori del formato

Descrittori di base:

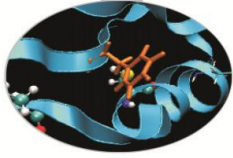
Tipo di dato	Descrittori
Integer	<i>Iw, Iw.m</i>
Floating point	<i>Ew.d, Ew.dEe, Fw.d, Gw.d, Gw.dEe</i>
Logical	<i>Lw</i>
Character	<i>A, Aw</i>

w = massimo numero di caratteri utilizzabili;

m = minimo numero di caratteri utilizzabili;

d = numero di cifre a destra del punto decimale;

e = numero di cifre dell'esponente.



L'istruzione READ

Le lettere maiuscole indicano:

I = numeri interi

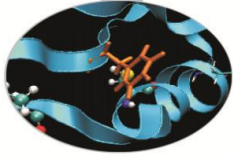
F = numeri reali

E = numeri nel formato esponenziale

L = dati logici, può indicare solo T (true) o F (false)

A = stringhe di caratteri

Descrittori del formato



Tra i descrittori di formato è possibile inserire:

/ = indica di passare a nuova riga (andare a capo)

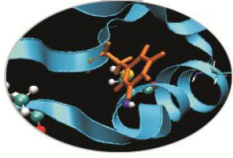
nX = sposta il cursore di n spazi bianchi.

```
WRITE (100, 10) "La media ottenuta e' ", val_med  
10 FORMAT (1X, A, F10.5)
```

oppure:

```
WRITE (100, "(1X, A, F10.5)") "La media ottenuta &  
&e' ", val_med
```

```
123456789.123456789.123456789.123456789.  
La media ottenuta e' -375.12345
```



Descrittori del formato

Esempio:

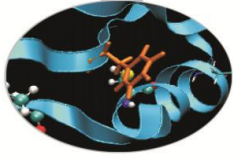
```
WRITE (100, 20) "I dati sono: ", &  
a,b,c, "La media ottenuta e' ", val_med
```

```
20 FORMAT (1X, A, 3(F10.5,1X), /, A, F10.5)
```

```
Write (100, 20) a,b,c
```

```
20FORMAT (f10.3)
```

Esercizi



5. Ripetere gli esercizi precedenti introducendo i descrittori di formato.