Operazioni base di Input e Output

Introduction to Fortran for Scientific Computing

Caterina Caravita
c.caravita@cineca.it

COSA VIENE PRESENTATO

- Input/Output
- Istruzioni fondamentali e sintassi:
  READ, WRITE, PRINT, OPEN, CLOSE
- Formattazione

Input/Output

Un programma comunica con l'esterno tramite **operazioni** e **unità** di **INPUT/OUTPUT** (**I/O**).

Generalmente un programma

- **legge** il valore delle variabili dall'esterno (**INPUT**)
- le **elabora**
- **scrive** i risultati verso l'esterno (**OUTPUT**)

Tutte le operazioni di I/O in Fortran fanno riferimento a unità di I/O: generalmente **tastiera**, **video**, **file su disco**, ma applicazioni specifiche possono prevedere altri tipi di unità di I/O (canali ottimizzati per la raccolta dati, plotter, stampanti, uscite audio...).

Prima di effettuare trasferimenti di dati, un file o un dispositivo devono essere preventivamente associati a una unità di I/O, rappresentate da un **numero intero**.

Input/Output

Operazioni fondamentali:

- `OPEN (FILE =“mydata.in”, UNIT=31)`
- `READ (UNIT=31,*) date`
- `WRITE (UNIT=31,*) results`
- `CLOSE (UNIT=31)`

Unità standard:

5, * --> std-in

6, * --> std-out

0, 3 --> std-err

Istruzione READ

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

Sintassi:

```
READ (elenco clausole) input
```

Esempio:

```
READ (*,*) n
```

Clausole principali:

- `UNIT = numero` (* = 5 -> std-in)
- `FMT = formato` (* -> formato libero, variable driven)
- `IOSTAT = valore` (opzionale)
- `END = etichetta` (opzionale)

Istruzione READ

Le **clausole** devono essere riportate con il loro nome.

```
READ (UNIT=41, FMT="(I4)", END=99, IOSTAT=ios) anno
```

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

```
READ (41, "(I4)", END=99) anno
```

```
READ (41, "(I4,2(I2))", END=99, IOSTAT=ios) aa, mm, gg
```

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:

```
READ (41, "(I4)") anno
```

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

```
READ (*, "(F10.2)", END=99, IOSTAT=ios) valore
```

Istruzione READ

La clausola **FMT** determina il **formato** dei dati da **leggere o scrivere**.

Il formato può essere specificato

- all'interno dell'istruzione

```
READ (41, "(8X, I4)") anno
```

- richiamando l'istruzione FORMAT tramite un'etichetta

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

- tramite una variabile precedentemente definita

```
CHARACTER (*), PARAMETER :: fmt1="(8X, I4)"
READ (*, fmt1) anno
```

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

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

Istruzione READ

La clausola **IOSTAT** ritorna un valore intero, tramite una variabile (precedentemente definita), indicativo dell'**esito** dell'**istruzione** (OPEN/READ/WRITE/CLOSE):

- `0` in caso di successo
- `/= 0` in caso di errore

```
READ (40, *,  IOSTAT=ios) valore
```

La clausola **END** permette di specificare l'etichetta a cui passare il **controllo** dell'esecuzione nel caso in cui la lettura giunga al termine del file (unità di input).

```
READ (40, *, END=99) valore
```

Istruzione WRITE

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

Sintassi:

```
WRITE (elenco clausole) output
```

Esempio:

```
WRITE (*,*) n
```

Clausole principali:

- `UNIT = numero` (* = 6 -> std-out)
- `FMT = formato` (* -> formato libero, variable driven)
- `IOSTAT = valore` (opzionale)
- `ERR = etichetta` (opzionale)

Istruzione WRITE

Le **clausole** hanno lo stesso significato e utilizzo delle clausole omonime dell'istruzione READ, escludendo END e con l'aggiunta di ERR.

La clausola **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.

```
WRTIE (UNIT=41, FMT="(I4)", ERR=99, IOSTAT=ios) anno
```

```
WRTIE (41, "(I4)", ERR=99, IOSTAT=ios) anno
```

Istruzione WRITE

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

WRITE (*, 100) anno
100 FORMAT ("Valore di ANNI: ", I4)

c = a + b
WRITE(*, "(2(I4,A),I4)") a, " + ", b, " = ", c
```

Istruzione PRINT

Solamente per la scrittura su **std-out**, in alternativa all'istruzione WRITE, esiste anche l'istruzione PRINT, con una sintassi più semplice.

Sintassi:

```
PRINT format, output
```

Clausola:

`FMT = formato` (* -> formato libero, variable driven)

Esempio:

```
PRINT *, n
```

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.

Istruzione OPEN

L'istruzione OPEN realizza un **collegamento** tra le aree disco e il programma, associando un determinato file ad un numero intero (unità di I/O).
Questo numero individua quel file in tutte le operazioni di I/O (OPEN/READ/WRITE/CLOSE).

Sintassi:

```
OPEN (elenco clausole)
```

Esempio:

```
OPEN (31, FILE="input.dat")
```

Clausole principali:

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

Istruzione OPEN

La clausola **UNIT** indica il numero di unità associato al file (in tutte le istruzioni di OPEN/READ/WRITE/CLOSE).

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

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

Istruzione OPEN

La clausola **IOSTAT** ha lo stesso significato nelle istruzioni READ/WRITE.

La clausola **STATUS** indica lo stato del file da aprire:

- `old` deve esistere già (utile per lettura; se vi si scrive, viene sovrascritto)
- `new` deve non esistere e viene creato (per scrittura)
- `replace` se non esiste, viene creato; se esiste, viene cancellato e ri-creato
- `scratch` file temporaneo (non bisogna specificare il nome del file)
- `unknown` old o new (default)

```
OPEN (UNIT=100, FILE="input.dat", IOSTAT=ios, STATUS="old")
```

Istruzione OPEN

La clausola **ACTION** indica la modalità con cui operare sul file:

- `read` solo lettura
- `write` solo scrittura
- `readwrite` lettura e scrittura (default)

```
OPEN (UNIT=100, FILE='input.dat', IOSTAT=ios, STATUS="old", ACTION="read")
```

Istruzione CLOSE

L'istruzione CLOSE **chiude** l'unità di I/O 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)
```

Esempio:

```
CLOSE (31)
```

Clausole principali:

- `UNIT = numero`
- `STATUS = stringa` (opzionale)
- `IOSTAT = valore` (opzionale)

Istruzione CLOSE

La clausola **UNIT** indica il numero di unità da chiudere (in tutte le istruzioni di OPEN/READ/WRITE/CLOSE).

La clausola **STATUS** indica lo stato del file dopo la chiusura:

- `keep` mantenere il file (default)
- `delete` rimuovere il file

```
CLOSE (UNIT=31, STATUS="delete")
```

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** (es. create un file ***Fahreneit.txt*** con alcuni valori di temperatura) e stampare i risultati a **video**.

```
Conversione Fahrenheit-Kelvin:
temp_k = (5. / 9.) * (temp_f + 459.67)

Conversione Kelvin-Celsius:
temp_c = temp_k - 273.15
```

Formato

I valori delle variabili sono memorizzati 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 può avvenire in diversi modi e viene comandata tramite l'istruzione "**FORMAT**".

Formato

Recap

Nelle istruzioni READ/WRITE, il formato può essere specificato

- all'interno dell'istruzione

```
READ (41, "(8X, I4)") anno
```

- richiamando l'istruzione FORMAT tramite un'etichetta

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

- tramite una variabile precedentemente definita

```
CHARACTER (*), parameter :: fmt1="(8X, I4)"
READ (*, fmt1) anno
```

Formato

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**).

```
READ (41, "2(8X, I4)") anno
```

Descrittori del formato

Tipo di dato	Descrittori
Integer	I*w*, I*w.m*
Floating point	E*w.d*, E*w.d*Ee, F*w.d*, G*w.d*, G*w.d*Ee
Logical	L*w*
Character	A, A*w*

- `I` numeri interi
- `F`, `G` numeri reali
- `E` numeri in formato esponenziale
- `L` dati logici (T - .true. o F - .false.)
- `A` stringhe di caratteri

Descrittori del formato

Tipo di dato	Descrittori
Integer	I*w*, I*w.m*
Floating point	E*w.d*, E*w.d*Ee, F*w.d*, G*w.d*, G*w.d*Ee
Logical	L*w*
Character	A, A*w*

- `w` massimo numero di caratteri utilizzabili (per tutti i tipi di variabili)
- `m` minimo numero di caratteri utilizzabili (per numeri interi)
- `d` numero di cifre decimali
- `e` numero di cifre dell'esponente

Descrittori di formato

Tra i descrittori di formato è possibile inserire:

`/` nuova riga
`nX` n spazi

```
WRITE (*, 20) "I dati sono: ", a, b, c, "Valore medio risultante: ", val_med
20 FORMAT (1X, A, 3(F10.5, 1x), /, A, F10.5)

WRITE (100, 20) a, b, c
20 FORMAT (f10.3)

WRITE (*, "(1X, A, F10.5)") "Valore medio &
                            risultante: ", val_med
```

ESERCIZI

- Ripetere gli esercizi precedenti introducendo i **descrittori di formato**.

GRAZIE

Caterina Caravita
c.caravita@cineca.it