

# Programmazione in Fortran:

## Lezione 5

A.A. 2009/2010

Ing. A . Siviglia

[nunzio.siviglia@ing.unitn.it](mailto:nunzio.siviglia@ing.unitn.it)

stanza:

Laboratorio didattico di modellistica idrodinamica (2° piano)

Tel 2440

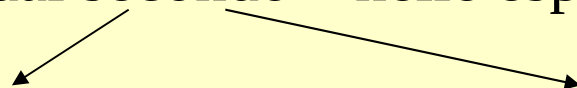


## Concetti di INPUT/OUTPUT

Finora abbiamo letto valori da tastiera e scritto sullo schermo usando le due istruzioni **READ** e **WRITE**.

Il formato che abbiamo usato e' detto **free-format** ed e' specificato dal secondo \* nelle espressioni

**READ ( \*, \* )** e **WRITE ( \*, \* )**



Come si e' visto, il risultato dell'utilizzo del free-format, fa apparire spazi non necessari.



# Formati e Formattazione

```
WRITE(*,100) i,risultato
```

```
100 FORMAT ('Iterazione numero',I3,1x,'vale', F7.3
```

L'istruzione FORMAT contiene le istruzioni per la formattazione utilizzata dalla istruzione WRITE

I3 e F7.3 sono descrittori di formato associati alle variabili i e risultato

Iterazione numero 3	vale 3.142	(formatted)
Iterazione numero	3 vale	3.141593 (unformatted)



# INTEGER output: il descrittore I

$rIw$  or  $rIw.m$

$r$  = **repeat count** numero di volte che deve essere usato il descrittore

$w$  = **Field width** numero di caratteri totali da usare

$m$  = Numero minimo di digits che devono essere visualizzati



# INTEGER output: il descrittore I

```
PROGRAM formati

  IMPLICIT NONE
  INTEGER:: index=-12 ,junk=4 ,number=-12345

  WRITE(*,200) index,index+12,junk,number
  WRITE(*,210) index,index+12,junk,number
  WRITE(*,220) index,index+12,junk,number

200 FORMAT(' ',2I5,  I6, I10)
210 FORMAT(' ',2I5.0, I6, I10.8)
220 FORMAT(' ',2I5.3, I6, I5)

END PROGRAM formati
```

```
      -12      0      4      -12345
      -12      4 -00012345
     -012    000    4*****
-----|-----|-----|-----|-----|
           5      10      15      20      25
```



# REAL output: il descrittore F

$rFw$     or     $rFw.d$

$r$  = **repeat count** numero di volte che deve essere usato il descrittore

$w$  = **Field width** numero di caratteri totali da usare

$d$  = Numero di digits a destra del punto decimale



# REAL output: il descrittore F

```
PROGRAM formati
```

```
  IMPLICIT NONE
```

```
  REAL    :: a = -12.3, b = .123 ,c = 123.456
```

```
  WRITE(*,200) a,b,c
```

```
  WRITE(*,210) a,b,c
```

```
200 FORMAT(' ',2F6.3,  F8.3)
```

```
210 FORMAT(' ',3F10.2)
```

```
END PROGRAM formati
```

```
***** 0.123 123.456
```

```
    -12.30
```

```
    0.12
```

```
   123.46
```

```
-----|-----|-----|-----|-----|-----|
```

```
    5
```

```
   10
```

```
   15
```

```
   20
```

```
   25
```

```
   30
```



# REAL output: il descrittore E

$rEw$  or  $rEw.d$

$r$  = **repeat count** numero di volte che deve essere usato il descrittore

$w$  = **Field width** numero di caratteri totali da usare

$d$  = Numero di digits a destra del punto decimale

Notazione esponenziale

4096.0 ---->  $0.4096 \times 10^4$

**Attenzione** il descrittore E normalizza i valori tra 0.1 e 1





# REAL output: il descrittore E

```
PROGRAM formati  
  
  IMPLICIT NONE  
  REAL    :: a = 1.2346E6, b = 0.001 , c = -77.7E10  
  
  WRITE(*,200) a,b,c  
  
200 FORMAT(' ',2E14.4,  E13.6)  
  
END PROGRAM formati
```

```
      0.1235E+07      0.1000E-02-0.777000E+12  
-----|-----|-----|-----|-----|-----|-----|-----|  
      5      10      15      20      25      30      35      40
```



# REAL output: il descrittore ES

$rESw$  or  $rESw.d$

$r$  = **repeat count** numero di volte che deve essere usato il descrittore

$w$  = **Field width** numero di caratteri totali da usare

$d$  = Numero di digits a destra del punto decimale

Notazione esponenziale

4096.0 ---->  $4.096 \times 10^3$

**Attenzione** il descrittore ES normalizza i valori tra 1.1 e 10



# REAL output: il descrittore ES

```
PROGRAM formati
```

```
  IMPLICIT NONE
```

```
  REAL    :: a = 1.2346E6, b = 0.001 , c = -77.7E10
```

```
  WRITE(*,200) a,b,c
```

```
200 FORMAT(' ',2ES14.4, ES13.6)
```

```
END PROGRAM formati
```

```
      1.2346E+06      1.0000E-03-7.770000E+11
-----|-----|-----|-----|-----|-----|-----|
      5      10      15      20      25      30      35      40
```



# CHARACTER output: il descrittore A

$rA$  or  $rAw$

$r$  = **repeat count** numero di volte che deve essere usato il descrittore

$w$  = **Field width** numero di caratteri totali da usare



# CHARACTER output: il descrittore A

```
PROGRAM formati
```

```
  IMPLICIT NONE
```

```
  CHARACTER(len=17)  :: string = 'This is a string'
```

```
  WRITE(*,10) string
```

```
  WRITE(*,11) string
```

```
  WRITE(*,12) string
```

```
10 FORMAT(' ',A)
```

```
11 FORMAT(' ',A20)
```

```
12 FORMAT(' ',A6)
```

```
END PROGRAM formati
```

```
This is a string
```

```
    This is a string
```

```
This i
```

```
-----|-----|-----|-----|-----|-----|-----|-----|
          5      10      15      20      25      30      35      40
```

# Posizionatori orizzontali: i descrittori X e T

$nX$

$n$  = numero di blanks da inserire

$Tc$

$c$  = numero della colonna a cui spostarsi



# Cambiare linea di output: il descrittore /

Il descrittore / fa saltare la scrittura alla linea successiva



## Ripetizione di gruppi di formato

```
320 FORMAT (1x, I6, I6, F10.2, F10.2, I6, F10.2, F10.2)
```

e' analogo a

```
320 FORMAT (1x, I6, 2(I6, 2F10.2))
```





# Descrittori di formato per la lettura (READ)

I descrittori di formato per la lettura sono analoghi a quelli per la scrittura WRITE.

L'interpretazione di tali descrittori è un po' differente.

Es:

```
READ(*, '(3F10.4)') a,b,c
```

dati di input sono

```
1.5      0.15E+01      15.0E-01
```

----> tutte le 3 variabili lette hanno lo stesso valore  
pari a 1.5



# FILE PROCESSING

I dati da processare sono immagazzinati all'interno di files. Si pone il problema di leggere questi dati, di processarli e di immagazzinare i risultati in un file nuovo.

Nelle istruzioni

`WRITE(*,*)` e `READ(*,*)`

asterisco nella prima posizione indica dove andare a scrivere e da dove leggere i dati.

  
`* ---->` indica la standard input/output device (sui pc sono la tastiera e il video).

Spesso vengono usati anche il 5 per indicare la tastiera e il 6 per il video

`WRITE(6,*)` ----> scrivi su video

`READ(5,*)` ----> leggi da tastiera



# ISTRUZIONE OPEN (apertura file)

OPEN (*open\_list*)

La *open\_list* e' una lista di opzione che contiene il numero dell'unita' da aprire, il nome del file, e informazioni su come accedere al file. Le singoli opzioni sono separate da una virgola.



# ISTRUZIONE OPEN (apertura file)

Le 5 piu' importanti opzioni per la lista dell'open sono:

- ✓ *unit* indica il numero dell'unita' associato al file da aprire

*unit* = numero intero

- ✓ *file* specifica il nome del file da aprire

*file* = e' un valore carattere



# ISTRUZIONE OPEN (apertura file)

Le 5 piu' importanti opzioni per la lista dell'open sono:

✓ *status* indica lo status del file da aprire

*status* = 'old'      *status* = 'new'

*status* = 'scratch'    *status* = 'unknown'

✓ *action* specifica se un file deve essere aperto solo in lettura,  
solo in scrittura o entrambi

*action* = 'read', *action* = 'write', *action* = 'readwrite',



# ISTRUZIONE OPEN (apertura file)

.....continua

✓ *iostat=inter\_var* indica il nome della variabile carattere in cui viene inserito un numero intero che dice se l'operazione di open e' stata eseguita correttamente o meno.

(*inter\_var* = 0 --> apertura/lettura corretta

*inter\_var* > 0 --> errore nella lettura o nel formato,

*inter\_var* < 0 ----> indica che e' stata raggiunta la fine del file)

ES:

```
INTEGER:: status
```

```
    iostat = status
```

```
OPEN(UNIT=8,FILE='esempio.dat',status='old',action='read', &
```

```
    iostat=status)
```



# ISTRUZIONE CLOSE (chiusura file)

Quando si finisce di utilizzare un file e' buona norma chiudere il file lasciando libera cosi' l'unita' di i/o associata

ES:

`CLOSE(unit)`



# ISTRUZIONE CLOSE (chiusura file)

Quando si finisce di utilizzare un file e' buona norma chiudere il file lasciando libera cosi' l'unita' di i/o associata

ES:

`CLOSE(unit)`





# Apertura di un file di lunghezza ignota

```
PROGRAM apertura_file_unknown
!  
!Questo codice legge i dati di un file di lunghezza non nota  
!  
  
    IMPLICIT NONE  
    CHARACTER(len=20)    :: nome_file  
    INTEGER:: lentrin,nvals = 0  ! numero di dati in ingresso  
    INTEGER:: status      !I/O status  
    REAL:: value  
  
    ! Acquisisce nome file da aprire e lo scrive sullo schermo  
    WRITE(*,*) 'Inserire nome del file da aprire'  
    READ(*,*) nome_file  
    ! nome_file = trim(nome_file)  
  
    WRITE(*,1000) nome_file  
1000 FORMAT(' ', 'nome file da aprire:', A)
```



```

!Apertura file e check per gli errori
  OPEN(UNIT=3,FILE=nome_file,STATUS='old',ACTION='read',IOSTAT=status)
  openif: IF (status == 0) THEN
!apertura file corretta --> leggi dati
    readloop: DO
      READ(3,*,IOSTAT=status) value ! Legge il dato
      IF (status /= 0) EXIT
      nvals = nvals +1
      WRITE(*,1010) nvals,value
1010   FORMAT(' ', 'Linea',I6,': Valore dato',F10.4)
    END DO readloop

! Il ciclo DO e' terminato. Perche' si e' raggiunta la fine del
! file o perche' c'e'e stato un errore?
!
    readif: IF(status > 0)THEN
      WRITE(*,1020) nvals+1
1020   FORMAT(' ', 'Si e verificato un errore alla linea',I6)
    ELSE ! si e raggiunta la fine del file
      WRITE(*,1030) nvals
1030   FORMAT(' ','Fine file raggiunta. Ci sono',1x,I6,1x,'dati nel file.')
    END IF readif

  ELSE openif

    WRITE(*,1040) status
1040   FORMAT(' ', 'Errore nell apertura del file: IOSTAT =',I6)

  END IF openif

!Close file
  CLOSE(3)

END PROGRAM apertura_file_unknown

```



## ESERCIZIO 1

Leggere 10 dati contenuti in un file (che dovete generare) e calcolare la media e la deviazione standard

## ESERCIZIO 2

Leggere gli n dati contenuti in un file di lunghezza incognita e calcolare la media e la deviazione standard

$$\hat{x} = \frac{1}{N} \sum_{i=1}^N x_i = \frac{x_1 + x_2 + x_3 + \dots + x_N}{N}$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \hat{x})^2}$$

