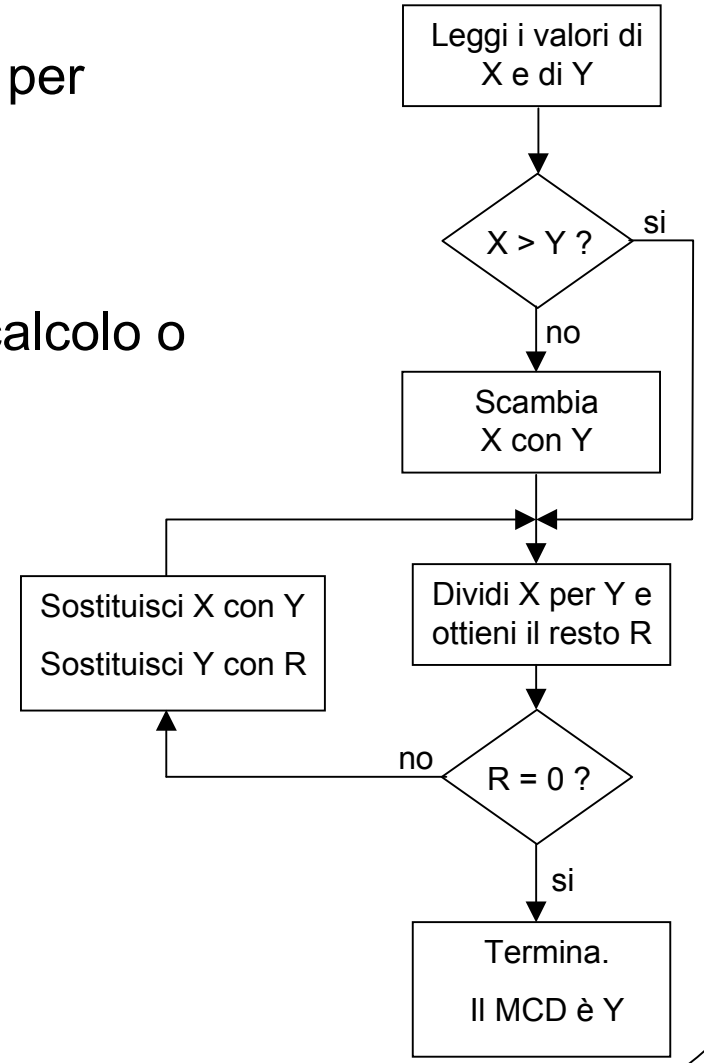
Classi di istruzioni

In maniera simile a quanto fatto per i dati, un linguaggio mette a disposizione dei costrutti per realizzare la parte esecutiva dell'algoritmo.

Questa consiste di:

- assegnazioni di valori a variabili (in base a calcolo o da I/O)
- selezione di azioni alternative in base alla valutazione di una condizione
- esecuzione ciclica di una o più azioni



I costrutti forniti dal linguaggio si dividono in corrispondenti **classi di istruzioni**

Le classi di istruzioni in Fortran

Istruzioni seriali
- Istruzione di calcolo e assegnazione
- Istruzioni di I/O

Istruzioni per il controllo di flusso
- Istruzioni selettive
  - IF
  - IF…ELSE
  - IF…ELSE IF
  - SELECT CASE
- Istruzioni per il controllo di ciclo
  - Cicli a conteggio (DO)
  - Cicli a condizione (DO IF…EXIT END DO,WHILE)

Istruzioni di calcolo e assegnazione

L'effetto è di aggiornare il valore di una variabile di un certo tipo con il valore ottenuto dalla valutazione di un'espressione dello stesso tipo.

Il formato è:

```
variabile = espressione
```

Esempio

```
INTEGER :: a,b
REAL :: x,y
LOGICAL :: cond

a = 4          x = 4.          cond = .TRUE.
b = 2*a+1      y = 2*b+x       cond = x > y
a = b          x = x-2*x*y     cond = (a>=0).AND.(a<=9)
a = a/2                        cond = cond .AND.(x==3.)
```

Istruzioni di Input/Output

Con le istruzioni di input, il valore di una variabile viene modificato con il valore ottenuto grazie ad un'operazione di lettura dall'unità di ingresso (tastiera).

Con le istruzioni di output, un'espressione viene valutata ed il valore ottenuto viene presentato sull'unità di uscita (monitor).

Il formato è:

```
READ(*,*) variabile
WRITE(*,*) espressione
```

Esempio

```
INTEGER :: a
REAL :: x

READ(*,*) a
READ(*,*) x
READ(*,*) 4        errato!
READ(*,*) 7*a+5    errato!
READ(*,*)
```

```
WRITE(*,*) a
WRITE(*,*) x
WRITE(*,*) 4
WRITE(*,*) 7*a+5
WRITE(*,*)
```

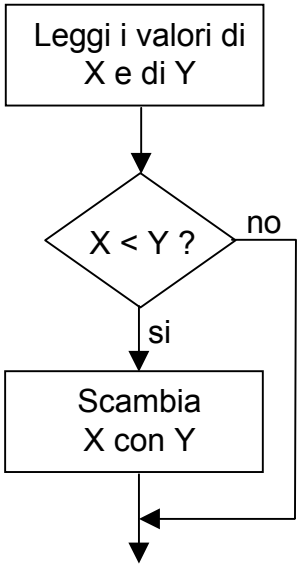
Istruzioni selettive: IF…THEN

Sintassi

```
IF (condizione) THEN
  istruzione_1
  istruzione_2
      ...
  istruzione_n
END IF
```

eseguite solo se *condizione* è TRUE

Esempio




```
INTEGER :: x,y,appo
READ(*,*) x,y
IF (x < y) THEN
  ! scambia x e y
  appo=x
  x=y
  y=appo
END IF
```

Istruzioni selettive: IF logico

Sintassi

IF (*condizione*) *istruzione*

Esempio

```
INTEGER :: x

READ(*,*) x
IF (x < 0) x=0
```

Istruzioni selettive: IF...THEN...ELSE

Sintassi

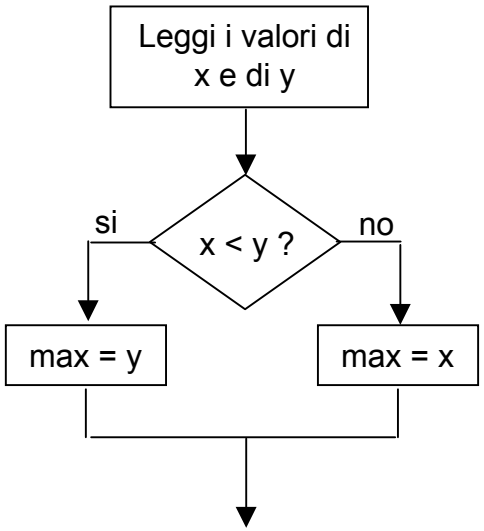
```
IF (condizione) THEN
  istruzione_1
  istruzione_2
ELSE
  istruzione_3
  istruzione_4
END IF
```

istruzione_1 e *istruzione_2*: eseguite solo se *condizione* è TRUE

istruzione_3 e *istruzione_4*: eseguite solo se *condizione* è FALSE

Esempio

Qual è il massimo tra x e y ?




```
REAL :: x,y,max
READ(*,*) x,y
IF (x < y) THEN
  max=y
ELSE
  max=x
END IF
```

Istruzioni selettive: IF...THEN...ELSE IF…ELSE

Sintassi

```
IF (condizione_1) THEN
  blocco_1
ELSE IF (condizione_2) THEN
  blocco_2
ELSE IF (condizione_3) THEN
  blocco_3
ELSE
  blocco_4
END IF
```

eseguito solo se *condizione_1* è TRUE

eseguito solo se *condizione_1* è FALSE e *condizione_2* è TRUE

eseguito solo se *condizione_1* è FALSE, *condizione_2* è FALSE e *condizione_3* è TRUE

eseguito solo se *condizione_1* è FALSE, *condizione_2* è FALSE e *condizione_3* è FALSE

Esempio

```
INTEGER :: voto
READ(*,*) voto
IF (voto < 18) THEN
  WRITE(*,*) “Ritorna”
ELSE IF (voto < 24)
  WRITE(*,*) “Si puo’ dare di piu’”
ELSE IF (voto < 27)
  WRITE(*,*) “Non c’e’ male”
ELSE IF (voto < 30)
  WRITE(*,*) “C’e’ mancato poco”
ELSE IF (voto == 30)
  WRITE(*,*) “Finalmente ci siamo”
ELSE
  WRITE(*,*) “WOW !”
END IF
```

Problema

Dati tre numeri reali x, y e z, stabilire se sono le lunghezze dei lati di un triangolo e, in caso affermativo, stabilire che tipo di triangolo sia (equilatero, isoscele o scaleno)

Istruzioni selettive: SELECT…CASE

Sintassi

```
CASE SELECT (espressione)
  CASE (selettore_1)
    blocco_1
  CASE (selettore_2)
    blocco_2
  CASE (selettore_3)
    blocco_3
  CASE DEFAULT
    blocco_4
END SELECT
```

espressione è di tipo intero

il *selettore* puo' indicare diverse condizioni:

`min:`	TRUE se *espressione* ≥ min
`min:max`	TRUE se min ≤ *espressione* ≤ max
`:max`	TRUE se *espressione* ≤ max
`val`	TRUE se *espressione* = val

Esempio

```
INTEGER:: voto

  WRITE(*,*) "Voto ottenuto:"
  READ(*,*) voto

  SELECT CASE (voto)
    CASE(0:17)
      WRITE(*,*) "Ritorna"
    CASE(18:24)
      WRITE(*,*) "Si puo' dare di piu'"
    CASE(25:27)
      WRITE(*,*) "Non c'e' male"
    CASE(28:29)
      WRITE(*,*) "C'e' mancato poco"
    CASE(30)
      WRITE(*,*) "Finalmente ci siamo"
    CASE(31:)
      WRITE(*,*) "WOW !"
    CASE DEFAULT
      WRITE(*,*) "Voto non valido"
  END SELECT
```

Istruzioni per il controllo di ciclo - ciclo a conteggio

Permette di ripetere l'esecuzione di un blocco di istruzioni per un numero predefinito di volte.

Sintassi

```
DO var_ciclo=espr_1,espr_2,espr_3
  istruzione_1
  istruzione_2
      ...
  istruzione_n
END DO
```

var_ciclo è una variabile numerica

espr_1, *espr_2* ed *espr_3* sono espressioni numeriche dello stesso tipo di *var_ciclo*

Esempio

stampare i numeri interi pari da 0 a 10

```
INTEGER :: i

DO i=0,10,2
  WRITE(*,*) i
END DO
```

Note

Il numero di iterazioni è valutato prima dell'esecuzione del ciclo ed è uguale a

MAX(INT((*expr_2* - *expr_1* + *expr_3*)/ *expr_3*),0)

La variabile di ciclo assume i valori compresi tra *expr_1* ed *expr_2* incrementandosi di *expr_3* ad ogni iterazione. Se *expr_3* è assente, viene assunto uguale a 1.

Se il numero di iterazioni è <= 0, le istruzioni in ciclo non vengono mai eseguite.

Il ciclo può realizzare anche conteggi decrescenti. In tal caso, *expr_2* < *expr_1* ed *expr_3* < 0.

Esempio

```
! 6 iterazioni
DO i=1,36,7
  ...
END DO
```

```
! 0 iterazioni
DO i=6,5
  ...
END DO
```

```
! 4 iterazioni
DO i=10,-3,-4
  ...
END DO
```

Problema

Stampare la “tabellina” di un valore dato in input.

Stampare le “tabelline” dei valori compresi tra 1 e 10.

Istruzioni per il controllo di ciclo - ciclo a condizione generica

Permette di ripetere l'esecuzione di un blocco di istruzioni finchè non viene verificata una condizione logica.

Sintassi

```
DO
   istruzione_1
      ...
   istruzione_k
      ...
   IF(condizione) EXIT
      ...
   istruzione_k+1
      ...
   istruzione_n
END DO
```

Problema

Leggere da input un insieme di numeri reali e calcolarne la media. Non si conosce in anticipo la quantità di valori da leggere, che comunque è limitata ad un massimo di 50; la lettura di un valore < 0 indica che l'insieme da leggere è terminato.

```
INTEGER :: cont
REAL :: x, somma, media
cont=0
somma=0.
DO
  WRITE(*,*) "Valore: "
  READ(*,*) x
  ! Condizione di uscita
  IF (x<0.) EXIT
  cont=cont+1
  somma=somma+x
  IF (cont>=50) EXIT
END DO
IF(cont>0)
  media=somma/cont
```

Note

Con questo costrutto, la verifica della condizione di uscita dal ciclo può essere posta in un punto qualunque del blocco. Inoltre, non è detto che sia unica.

Sono caratteristiche positive ?

Non è facile individuare nel ciclo le condizioni di uscita. Codice non facilmente comprensibile.

A tempo di esecuzione non è desumibile per quale condizione il ciclo è terminato e quindi quali istruzioni sono state eseguite. Incertezza sul punto di uscita e sulla condizione di uscita dal ciclo

Quindi...

È meglio avere un'unica condizione di uscita.

È meglio avere una condizione di uscita all'inizio o alla fine del costrutto.

Esempio: come ristrutturare un ciclo

```
cont=0
somma=0.

DO
  WRITE(*,*) "Valore: "
  READ(*,*) x
  ! 1a Condizione di uscita
  IF (x<0.) EXIT
  cont=cont+1
  somma=somma+x
  ! 2a Condizione di uscita
  IF (cont>=50) EXIT
END DO
```

```
cont=0
somma=0.

WRITE(*,*) "Valore: "
READ(*,*) x
DO
  IF (x<0.).OR.(cont>=10) EXIT
  cont=cont+1
  somma=somma+x
  WRITE(*,*) "Valore: "
  READ(*,*) x
END DO
```

Istruzioni per il controllo di ciclo - ciclo WHILE

Permette di ripetere l'esecuzione di un blocco di istruzioni finchè viene verificata una condizione logica valutata all'inizio del ciclo.

Sintassi

```
DO WHILE(condizione)
  istruzione_1
  istruzione_2
     ...
  istruzione_n
END DO
```

E' equivalente a:

```
DO
  IF (.NOT. condizione) EXIT
  istruzione_1
  istruzione_2
     ...
  istruzione_n
END DO
```

Esempio di utilizzo del WHILE

```
cont=0
somma=0.

WRITE(*,*) "Valore: "
READ(*,*) x
DO
  IF (x<0.).OR.(cont>=50) EXIT
  cont=cont+1
  somma=somma+x
  WRITE(*,*) "Valore: "
  READ(*,*) x
END DO
```

```
cont=0
somma=0.

WRITE(*,*) "Valore: "
READ(*,*) x
DO WHILE ((x>=0.).AND.(cont<50))
  cont=cont+1
  somma=somma+x
  WRITE(*,*) "Valore: "
  READ(*,*) x
END DO
```

Problema 1

Leggere da input un insieme di numeri reali >= 0 e determinare il valore massimo. Non si conosce in anticipo la quantità di valori da leggere; la lettura di un valore < 0 indica che l'insieme da leggere è terminato.

Problema 2

Nelle stesse ipotesi del problema 1, determinare il valore minimo dell'insieme dei valori letti.

Un programma da realizzare utilizzando un ciclo a condizione