Costrutti condizionali e iterativi

Introduction to modern Fortran

Moreno Guernelli
m.guernelli@cineca.it

COSA VIENE PRESENTATO

- Uscita e salti:
  STOP, GO TO
- Istruzioni condizionali:
  IF, CASE
- Cicli iterativi:
  DO, DO WHILE

Strutture di controllo

Le strutture di controllo permettono di **alterare la sequenza** di esecuzione delle istruzioni del programma al verificarsi di determinate **condizioni**.

STOP

L'istruzione STOP implica l'**interruzione** del programma e il ritorno al sistema operativo, eventualmente stampando a video un messaggio.

```
STOP 'messaggio'
```

Buona pratica prima di terminare il programma:

```
...
STOP
END PROGRAM
```

GO TO

L'istruzione GO TO permette di **saltare** da una riga ad un'altra del programma, tramite un'etichetta.

È un'istruzione sconsigliata!

GO TO **non condizionale**:

```
GO TO etichetta
```

Esempio:

```
GO TO 30
30 ...
```

GO TO **condizionale** (o **calcolato**):
Si salta all'n-esima etichetta in funzione del valore intero valutato
(se l'intero vale 1 si salta alla label1, …)

```
GO TO (label1, label2, …, labeln) espressione_intera
```

Costrutto IF

Il costrutto IF è una **struttura di selezione**, che permette di eseguire una sequenza di comandi **se** la corrispondente espressione logica è **vera**.

L'**indentazione** è molto utile, ma non obbligatoria.

Sintassi:

```
IF (condizione_logica) THEN
    sequenza di istruzioni
END IF
```

Opzionalmente, si può dare un nome alla struttura:

```
nome: IF (condizione_logica) THEN
    sequenza di istruzioni
END IF nome
```

Costrutto IF

La struttura **IF - ELSE IF - ELSE** permette di esprimere più condizioni logiche. Una volta che una condizione è stata trovata valida, viene eseguita la corrispondente sequenza di comandi, al temine della quale si esce dall'intero blocco IF.

Sintassi:

```
IF (condizione_logica1) THEN
    sequenza di istruzioni
ELSE IF (condizione_logica2) THEN
    sequenza di istruzioni
ELSE
    sequenza di istruzioni
END IF
```

Costrutto IF

Se il costrutto IF contiene una sola condizione logica e un'istruzione breve può essere scritto **in riga** (senza THEN e END IF).

Sintassi:

```
IF (condizione_logica) istruzione
```

Costrutto IF

Esempio:

```
IF (numero < 0) THEN
   mia_stringa = "numero negativo"
ELSE IF (numero == 0) THEN
   mia_stringa = "nullo"
ELSE
   mia_stringa = "numero positivo"
END IF
```

Costrutto CASE

Anche il costrutto CASE rappresenta una **struttura di selezione**, similarmente all'IF. Le istruzioni di un CASE vengono svolte a seconda che il valore di **una stessa** espressione rientri nel range di un certo selettore.

Sintassi:

```
nome: SELECT CASE (espressione)
   CASE (selettore1)
     istruzioni
   CASE (selettore2)
     istruzioni
   CASE DEFAULT
     istruzioni
   END SELECT nome
```

Costrutto CASE

L'espressione che guida l'esecuzione delle istruzioni del costrutto può essere qualunque espressione a **valore numerico intero**, di **carattere** o di tipo **logico**.

Il selettore può essere specificato da un **valore singolo**, da un **intervallo di valori** o da una **lista di valori**.

Sintassi:

```
nome: SELECT CASE (espressione)
  CASE (selettore1:selettore2)
    istruzioni
  CASE (selettore3, selettore4)
    istruzioni
  CASE DEFAULT
    istruzioni
  END SELECT nome
```

Costrutto CASE

Esempi:

```
tipi: SELECT CASE (carattere)
  CASE ('A':'Z', 'a':'z')
    tipo = "lettera"
  CASE ('0':'9')
    tipo = "cifra"
  CASE DEFAULT
    tipo = "simboli"
  END SELECT tipi
```

```
distingue: SELECT CASE (numero)
  CASE (:-1)
    mia_stringa = "numero negativo"
  CASE (1:)
    mia_stringa = "numero positivo"
  CASE DEFAULT
    mia_stringa = "nullo"
  END SELECT distingue
```

Differenza tra CASE e IF

L'istruzione IF viene utilizzata per valutare **condizioni logiche** e può essere annidata o estesa usando ELSE IF ed ELSE, ideale per situazioni in cui le condizioni da verificare sono varie e di natura logica, non necessariamente legate al valore di una singola variabile.

Il costrutto CASE, tramite SELECT CASE, è più indicato quando bisogna eseguire codici diversi in base **al valore assunto da una variabile** (tipicamente intera, carattere o logica). La sintassi è più leggibile rispetto a molteplici IF annidati e migliora la manutenzione del codice quando le alternative sono molte e si riferiscono a diversi valori della stessa variabile.

Ciclo DO

Il costrutto DO permette di **ripetere iterativamente** un insieme di istruzioni **finché** una certa condizione è **vera**.
È un **ciclo definito**: Il numero di ripetizioni è noto prima dell'inizio del ciclo.

Clausola iterativa: il contatore assume ad ogni ciclo un valore tra inizio e fine (**inclusi**), saltando di un passo (default 1); queste variabili devono essere tutte numeri **interi**.

Sintassi:

```
nome: DO contatore = inizio, fine, passo
   istruzioni
END DO nome
```

Esempio:

```
Somma: DO i = 2, 100, 2
   a = a + i
   print *, a
END DO Somma
```

Ciclo DO

Anche un ciclo DO può essere scritto **in riga**:

```
((expr(i,j), i=inizio,fine,step), j=inizio,fine,step)
```

Ad esempio, in operazioni di I/O

```
WRITE (*,10) (elenco(i), i=1,10)
10 FORMAT ("elenco= ", 10F8.5)
```

o nell'inizializzazione di un vettore

```
a = [(i*2, i=1,N)]
```

Ciclo DO WHILE

La clausola WHILE permette di **ripetere iterativamente** un insieme di istruzioni, **finché** una certa condizione logica è **vera**.
È un **ciclo indefinito**: il numero di ripetizioni non è noto prima dell'inizio del ciclo.

Sintassi:

```
nome: DO WHILE (espressione_logica)
   istruzioni
END DO nome
```

Esempio:

```
x = x0
Somma: DO WHILE (x < x1)
   a = funz(x)
   x = x + a
END DO Somma
```

Ciclo DO con CYCLE e EXIT

L'istruzione CYCLE permette di passare direttamente al **ciclo successivo**.
L'istruzione EXIT provoca l'**uscita** dal blocco iterativo.

Il costrutto DO può essere espresso anche **senza clausola**: in questo caso le istruzioni CYCLE e EXIT regolano le iterazioni.

Sintassi:

```
nome: DO
   istruzioni
   IF (espressione_logica1) CYCLE
   IF (espressione_logica2) EXIT
   Istruzioni
END DO nome

Istruzioni
```

Passa il controllo a END DO. Quindi passiamo al contatore successivo.

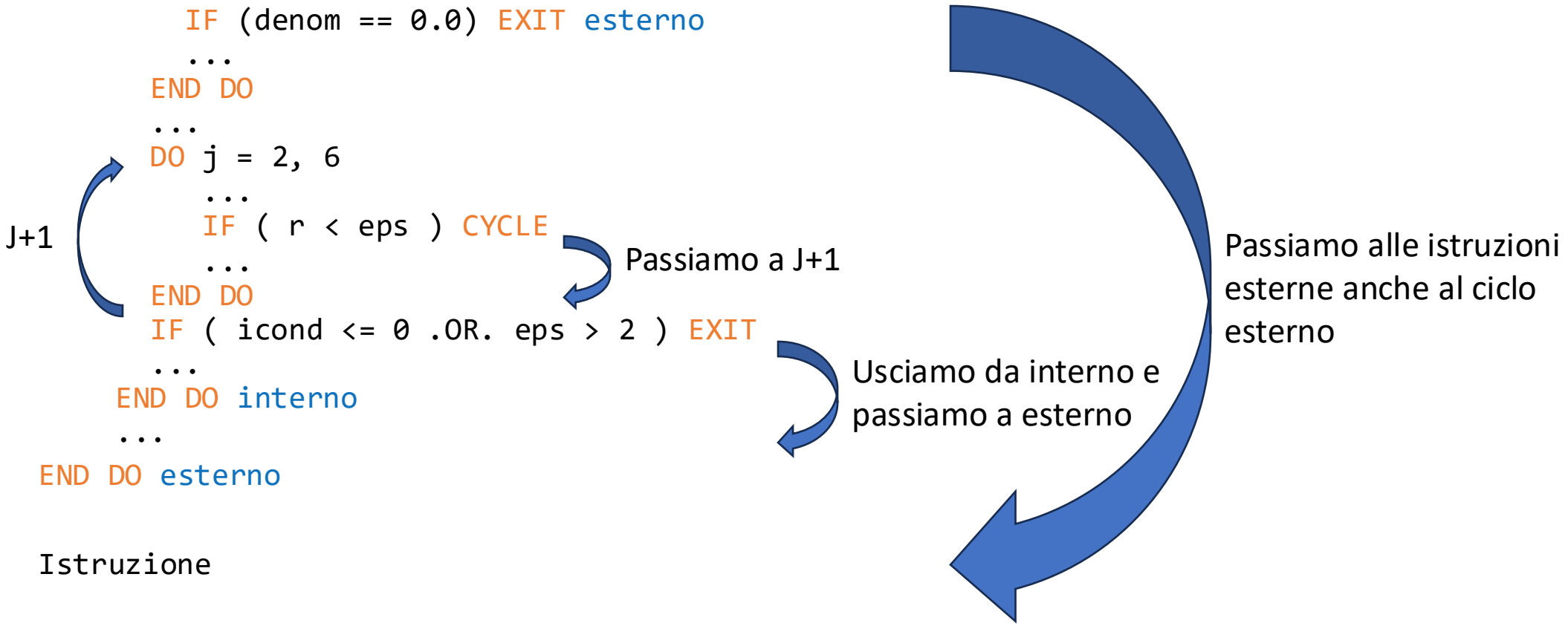
Passa il controllo all'istruzione successiva a END DO. Usciamo dal ciclo

Ciclo DO

Esempio:

```
esterno: DO
     interno: DO WHILE (icond > 0)
       DO i = 1, 10
         ...
         IF (denom == 0.0) EXIT esterno
         ...
       END DO
       ...
       DO j = 2, 6
          ...
          IF ( r < eps ) CYCLE
          ...
       END DO
       IF ( icond <= 0 .OR. eps > 2 ) EXIT
       ...
     END DO interno
     ...
END DO esterno

Istruzione
```



ESERCIZI

1. Scrivere un programma che stampa il maggiore tra due numeri interi, facendo uso del costrutto **IF**.

2. Scrivere un programma per la classificazione dei triangoli (equilatero, isoscele, scaleno), facendo uso del costrutto **IF**.

3. Scrivere un programma che calcola le soluzioni di un'equazione di secondo grado, facendo uso del costrutto **IF**.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

4. Scrivere un programma, contenente un ciclo **DO**, che legge numeri reali da input (es. creare un file ***square.dat*** con alcuni numeri), salta i numeri negativi, si interrompe se legge zero, somma le radici quadrate dei numeri positivi (usare **EXIT** e **CYCLE**).

ESERCIZI

5. Scrivere un programma che, dato un numero intero n, calcoli i valori della tavola pitagorica da 1 a n, stampando un prodotto per riga.

6. Scrivere un programma contenente un costrutto **CASE** che calcola il numero di giorni di un dato mese (leggere mese e anno).

7. Scrivere un programma che, dato un giorno, determina la data del giorno successivo.

8. Scrivere un programma che converte il testo da maiuscolo in minuscolo e viceversa.
   Suggerimento: usare Tabella ASCII.

9. Usando il costrutto **CASE**, scrivere un programma che operi la conversione in numeri romani dei numeri compresi tra 0 e 999.
   Suggerimento: salvare il numero romano come stringa di caratteri.

GRAZIE

Moreno Guernelli
m.guernelli@cineca.it