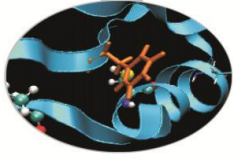


# Costrutti condizionali e iterativi

***Introduction to Fortran 90***

**Paolo Ramieri, CINECA**

*Aprile 2014*

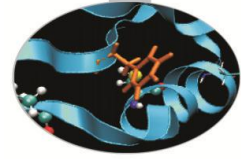


# Strutture di controllo

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

Due tipi fondamentali di istruzioni di controllo:

- IF: Istruzioni Condizionali
- DO LOOP: Cicli

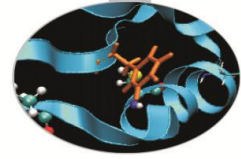


# Il costrutto IF

Questa struttura si occupa di eseguire una sequenza di comandi **solo quando la corrispondente espressione logica è vera.**

## Sintassi:

```
nome: IF (condizione logica) THEN  
    sequenza di comandi  
END IF nome
```

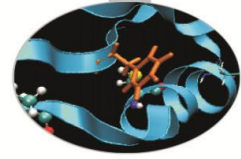


# Il costrutto IF

**IF con più condizioni logiche.** Una volta che **una condizione è stata trovata valida**, viene eseguita la corrispondente sequenza di comandi, **al termine della quale si esce dall'intero blocco IF.**

## Sintassi:

```
nome: IF (espressione_logica) THEN
    sequenza di comandi
ELSE IF (espressione_logica) THEN
    sequenza di comandi
ELSE
    sequenza di comandi
END IF nome
```

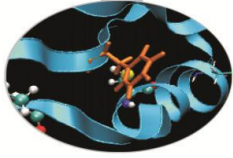


# Il costrutto IF

Il costrutto `IF` può riferirsi anche ad **una sola espressione logica**, in questo caso si può omettere l'istruzione `THEN` e `END IF`.

## Sintassi:

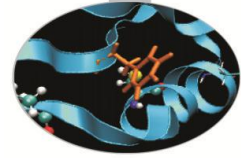
```
IF (espressione logica) comando_da_eseguire
```



# Il 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
```

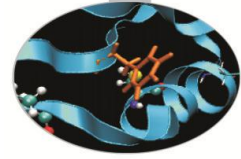


# Il costrutto CASE

Come il costrutto `IF/ELSE`, `CASE` rappresenta una struttura di selezione. Le istruzioni tra un `CASE` e l'altro vengono svolte a seconda che il valore di **espressione** rientri nel range di un certo **selettore**.

## Sintassi:

```
nome: SELECT CASE (espressione)  
    CASE (selettore)  
        istruzioni  
    CASE (selettore)  
        istruzioni  
    CASE DEFAULT  
        istruzioni  
END SELECT nome
```



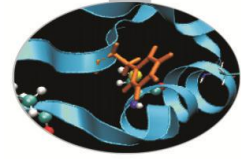
# Il costrutto CASE

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 una lista di valori o da un'estensione di valori.

## Sintassi:

```
test1: SELECT CASE (value)  
  CASE (1)  
    istruzioni  
  CASE (2)  
    istruzioni  
  CASE DEFAULT  
    istruzioni  
END SELECT test1
```

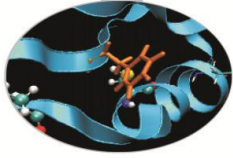




# Il costrutto CASE

## Esempio:

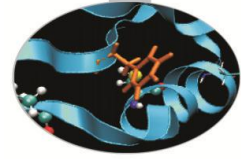
```
tipi: SELECT CASE (carattere)
  CASE ('A':'Z', 'a':'z')
    tipo = "lettera"
  CASE ('0':'9')
    tipo = "cifra"
  CASE DEFAULT
    tipo = "simboli"
END SELECT tipi
```



# Istruzioni di ciclo

Permettono di ripetere un insieme di istruzioni finchè una certa condizione si verifica.

- **Cicli definiti o iterativi:** Il numero di ripetizioni è noto prima dell'inizio del ciclo .
- **Cicli indefiniti:** Il numero di ripetizioni non è noto in anticipo.



# Il costrutto DO

DO con clausola iterativa:

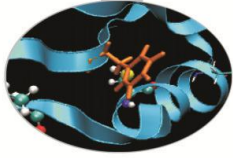
La variabile intera `step` determina il passo con cui devono essere eseguite le iterazioni; il valore di default è 1.

## Sintassi:

```
Nome: DO index = start, end, step
      istruzioni
      END DO nome
```

## Esempio:

```
Somma: DO i = 1, 1000, 1
        a=a+i
      END DO Somma
```



# Il costrutto DO

DO con clausola WHILE:

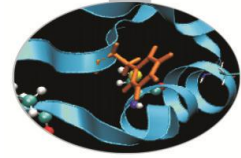
## Sintassi:

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

## Esempio:

```
i=0
```

```
Somma: DO WHILE (i<1000)
        i=i+1
        a=a+i
      END DO Somma
```

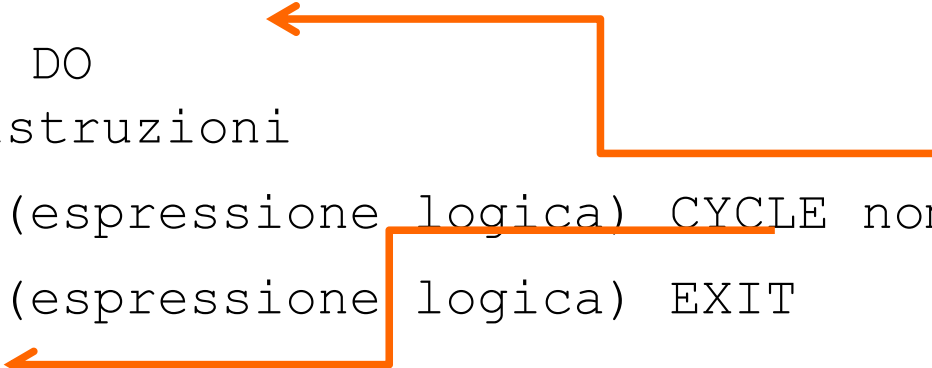


# Il costrutto DO

DO senza clausola: l'uscita dal blocco DO è affidata alle istruzioni CYCLE e EXIT.

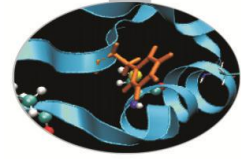
## Sintassi:

```
nome: DO
    istruzioni
    IF (espressione logica) CYCLE nome
    IF (espressione logica) EXIT
END DO nome
```



L'istruzione CYCLE permette di passare direttamente al ciclo successivo.

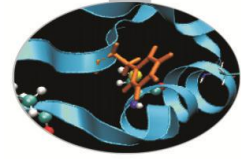
L'istruzione EXIT provoca l'uscita dal blocco iterativo.



# Il costrutto 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
```



# Il costrutto DO

DO implicito (utile nelle operazioni di I/O):

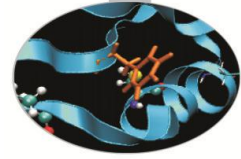
## Sintassi:

```
READ (unità, formato) (val(i), i=inizio, fine, step)
```

```
WRITE (unità, formato) (val(i), i=inizio, fine, step)
```

## Esempio:

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



# Il costrutto FORALL

Questo costrutto permette di esprimere più efficacemente che con i blocchi DO le operazioni da effettuarsi su vettori e matrici.

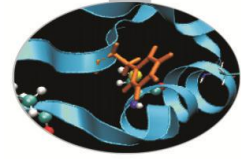
## Sintassi:

```
FORALL (i=m1:n1:k1, . . . . , j=m2:n2:k2, MASK)  
    istruzioni  
END FORALL
```

## FORALL con una sola istruzione:

```
FORALL (i=m1:n1:k1, . . . . , j=m2:n2:k2, MASK) &  
    x(i, . . . . , j) = espressione
```





# Il costrutto FORALL

Contrariamente allo spirito del Fortran 90 con il costrutto FORALL vettori e matrici sono viste ancora come insiemi di elementi.

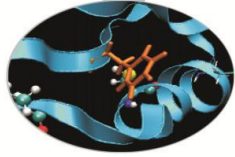
## Esempi:

```
FORALL (i=1:n) a(i,i)=i
```

```
FORALL (i=1:n, j=1:n, y(i,j) /= 0 .AND. i /= j) &  
  x(i,j) = 1.0 / y(i,j)
```

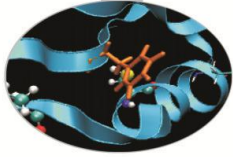
```
FORALL (i=1:n)  
  a(i,i) = i  
  b(i) = i * i  
END FORALL
```

# Esercizi



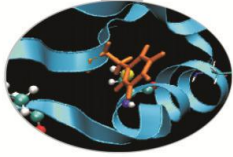
1. Scrivere un programma che stampa il maggiore tra due numeri interi.
2. Scrivere un programma per la classificazione dei triangoli (equilatero, isoscele, scaleno) facendo uso del costrutto IF.
3. Calcolo delle soluzioni di un'equazione di secondo grado.
4. Scrivere un programma contenente un ciclo DO che legge numeri reali da input, salta i numeri negativi, si interrompe se legge zero, somma la radice quadrata 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.

# Esercizi



8. Scrivere un programma che converte il testo da maiuscolo in minuscolo e viceversa.
9. Conversione da numero decimale a numero romano: usando il costrutto `SELECT 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.