

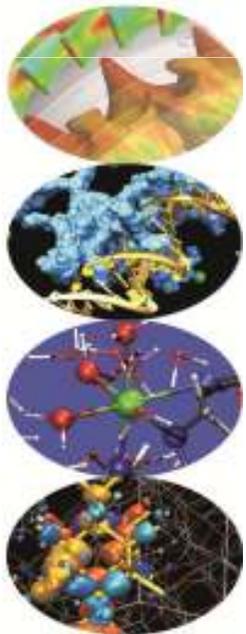


Nuovi operatori

Introduction to Fortran 90

Nicola Spallanzani, CINECA

Paolo Ramieri, CINECA



Ottobre 2014



Definizione

La sintassi per generare **nuovi operatori** è la seguente:

```
INTERFACE OPERATOR (.nuovo_operatore.)  
    Interfaccia del nuovo operatore  
END INTERFACE
```

Il **nome dell'operatore** è contraddistinto dall'aver il **punto (.)** all'inizio e alla fine.



Definizione

Conviene definire i nuovi operatori in un **modulo**.

In questo caso l'interfaccia ha la sintassi:

```
INTERFACE OPERATOR(.nuovo_operatore.)  
    MODULE PROCEDURE proc1  
END INTERFACE
```



Precisazioni

Un nuovo operatore:

1. deve essere implementato con una **funzione** ad argomenti non opzionali
2. il nome dell'operatore deve essere composto dai **solli caratteri alfabetici**
3. può essere **monadico** o **diadico**

Operatore diadico



```
INTERFACE OPERATOR (.distanza.)  
  MODULE PROCEDURE calcdist  
END INTERFACE
```

```
REAL FUNCTION calcdist (px,py)  
  IMPLICIT NONE  
  TYPE (punto), INTENT(IN) :: px, py  
  .  
  .  
  .  
END FUNCTION calcdist
```

```
d = p1.distanza. p2
```

Operatore monadico



```
INTERFACE OPERATOR (.normaII.)  
    MODULE PROCEDURE calcdist0  
END INTERFACE  
  
REAL FUNCTION calcdist0 (px)  
    IMPLICIT NONE  
    TYPE (punto), INTENT(IN) :: px  
        .      .      .  
END FUNCTION calcdist  
  
n2 = .normaII. px
```



Istruzione SEQUENCE

- Garantisce l'esatto ordinamento in memoria delle entità che compongono il tipo derivato
- Se il tipo personalizzato è composto da altri tipi personalizzati, è allora necessario che anche questi contengano l'istruzione SEQUENCE.



Uso delle interfacce

Casi in cui è necessario usare i blocchi interfaccia.

- A. Nel caso di una procedura *esterna* o una procedura *interna ad un modulo*:
1. per definire un operatore nuovo o estenderne uno predefinito;
 2. per definire un nome di procedura generico



Uso delle interfacce

B. Nel caso di una procedura *esterna*:

1. se gli argomenti sono richiamati con ordine libero
2. se c'è un argomento opzionale
3. se ci sono procedure esterne passate come argomento
4. se è una funzione vettoriale
5. se si utilizzano nomi generici
6. se è una funzione di tipo POINTER
7. se è una funzione di tipo CHARACTER di lunghezza nè costante nè presunta
8. se ha un argomento di tipo vettore a dimensioni presunte o di tipo POINTER o TARGET



Esercizi

1. Si provi a completare la definizione dell'operatore .distanza, aggiungendo il codice mancante.

Si ricorda che la distanza tra 2 punti P, Q è definita come

$$\text{SQRT}((P(1) - Q(1)) ** 2 + (P(2) - Q(2)) ** 2)$$

Proposte di soluzioni



Esercizio 1:

```
MODULE punti2D
  IMPLICIT NONE

  TYPE punto
    REAL(8) :: x, y
  END TYPE punto

  INTERFACE OPERATOR (.distanza.)
    MODULE PROCEDURE calcdist
  END INTERFACE
```



Proposte di soluzioni

CONTAINS

```
FUNCTION calcdist (p,q)
  IMPLICIT NONE
  TYPE (punto), INTENT(IN) :: p, q
  REAL(8) :: calcdist

  calcdist = SQRT((p%x-q%x)**2 + (p%y-q%y)**2 )

  RETURN
END FUNCTION calcdist
END MODULE punti2D
```



Proposte di soluzioni

```
PROGRAM distanze
  USE punti2D
  IMPLICIT NONE
  TYPE (punto) :: a, b
  REAL(8) :: d, n

  PRINT*, "Coordinate del punto A (x, y)"
  READ*, a
  PRINT*, "Coordinate del punto B (x, y)"
  READ*, b

  d = a .distanza. b
  PRINT*, "La distanza tra A e B e': ", d

END PROGRAM distanze
```