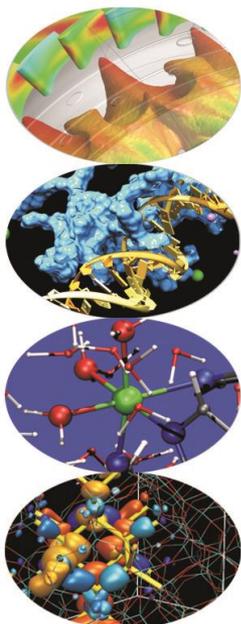
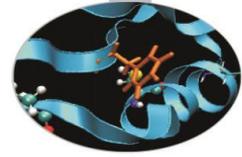


# Tipi personalizzati

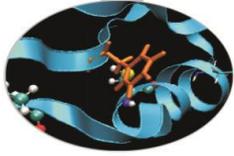




# Tipi personalizzati

## A COSA SERVONO:

- permettono di raggruppare in un'unico oggetto dati eterogenei;
- possono essere usati con operatori nuovi e operatori intrinseci;
- la realizzazione di tipi personalizzati e la ridefinizione degli operatori costituiscono il punto di partenza per una programmazione orientata agli oggetti.



# Tipi predefiniti

**Richiamo**: dichiarazione di tipi predefiniti

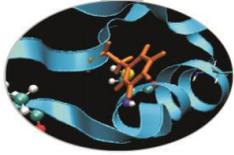
```
INTEGER :: i, j, k, ivett(10)
```

```
REAL(8) :: a(10,10), b(10,20)
```

```
COMPLEX(16) :: z, w, x(100,100)
```

```
CHARACTER :: intro = "La modalita' e': "
```

```
LOGICAL :: vero = .T., falso = .F., cond
```



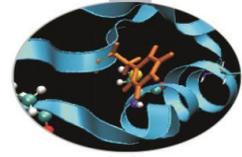
# Tipi personalizzati

## Sintassi per la definizione:

```
TYPE data
    INTEGER :: g, m, a
END TYPE data
```

Possono contenere diverse componenti di vario tipo;

```
TYPE nascita
    CHARACTER(132) :: luogo
    INTEGER :: g, m, a
END TYPE
```



# Tipi personalizzati

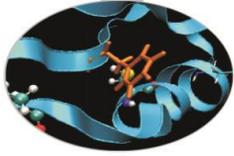
## Sintassi per la definizione:

Sono ammessi i tipi derivati ricorsivi

```
TYPE nascita
    CHARACTER(132) :: luogo
    TYPE(data) :: giorno
END TYPE nascita
```

## Sintassi per la dichiarazione:

```
TYPE(nascita) :: a, b, c
```



# Tipi personalizzati

## Sintassi per l'inizializzazione:

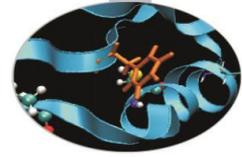
```
TYPE (nascita) :: a, b, c
```

```
a%luogo = "NewYork"
```

```
a%giorno%g = 16
```

```
a%giorno%m = 8
```

```
a%giorno%a = 1998
```

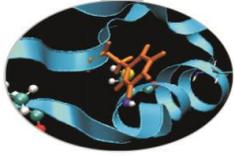


# Tipi personalizzati

## Assegnazione:

Tra gli operatori predefiniti solo l'assegnazione è applicabile ai tipi personalizzati:

```
TYPE data
    INTEGER :: g, m, a
END TYPE data
TYPE nascita
    CHARACTER(132) :: luogo
    TYPE(data) :: giorno
END TYPE nascita
TYPE(nascita) :: n
n = nascita("Citta' del Capo",data(29,2,2004))
```

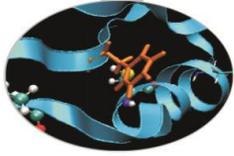


# Tipi personalizzati

## Stampa:

Il Fortran gestisce direttamente la presentazione dei dati relativi ai tipi personalizzati:

```
TYPE data
    INTEGER :: g, m, a
END TYPE data
TYPE nascita
    CHARACTER(132) :: luogo
    TYPE(data) :: giorno
END TYPE nascita
TYPE(nascita) :: n
n = nascita("Citta' del Capo", data(29, 2, 2004))
PRINT*, "Sono nato a ", n
```

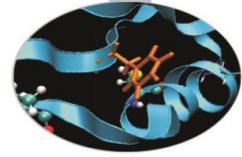


# Dati strutturati

## Costruttori (Fortran 2003)

Il Fortran 2003 estende la sintassi per la definizione dei tipi personalizzati.

In particolare non è più necessario specificare tutti i valori, ma solo nel caso i membri del tipo derivato siano stati inizializzati.



# Dati strutturati

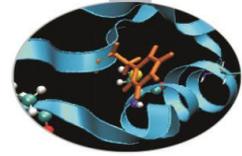
## Costruttori (Fortran 2003)

Ad esempio, se il tipo derivato *punto* viene così definito:

```
TYPE punto
    REAL(8) :: x=0, y=0    ! inizializzazione
END TYPE punto
TYPE segmento
    TYPE(punto) :: a, b
END TYPE segmento
```

Si può inizializzare il segmento *s* passando esplicitamente il valore solo del primo membro:

```
s = segmento(punto(x, y))    ! Non è
                             ! indispensabile passare esplicitamente
                             ! il secondo punto, se è (0,0)
```

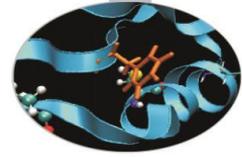


# Dati strutturati

## Costruttori (Fortran 2003)

Il Fortran 2003 permette inoltre di definire, in un tipo derivato, componenti pubbliche (accessibili all'esterno del modulo in cui è definito) e componenti private (non accessibili al di fuori del modulo). Esempio:

```
MODULO Geom2D
  TYPE punto
    REAL(8) :: x=0, y=0    ! Inizializzazione
    INTEGER, PRIVATE :: nd = 2 ! Non accessibile fuori dal
                                ! codice esterno al modulo Geom2D
  END TYPE punto
  TYPE segmento
    TYPE(punto) :: a, b
  END TYPE segmento
END MODULE Geom2D
```



# Dati strutturati

## Costruttori (Fortran 2003)

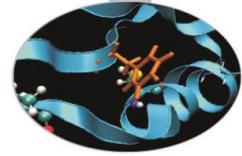
L'assegnazione di un valore al segmento  $s$  con il costruttore nella sintassi 2003 è possibile anche in questo caso; ovviamente non è possibile assegnare esplicitamente un valore alla componente privata  $nd$ :

```
USE geom
```

```
...
```

```
x = 1; y = 1; w = 2; z = 3;
```

```
s = segmento (punto (x, y), punto (w, z))
```



# Dati strutturati

## Costruttori (Fortran 2003)

Inoltre, anche al costruttore è possibile passare i valori dei membri in ordine diverso da quanto definito nel tipo derivato, purché se ne indichi il nome:

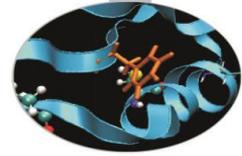
```
USE geom
```

```
...
```

```
x = 1; y = 1; w = 2; z = 3;
```

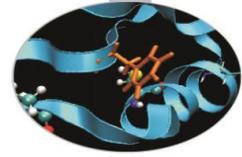
```
s = segmento(b = punto(y = w))
```

Nell'esempio al primo membro di segmento ( $a$ ) non viene assegnato un valore esplicitamente, perciò il valore di default (0,0) viene considerato; nel secondo membro ( $b$ ) viene assegnato esplicitamente il valore alla sola componente  $y$ .



# Esercizi

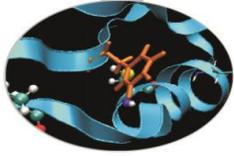
1. Dato il tipo nascita, come dagli esempi precedenti, si scriva un programma che, dopo aver definito il tipo “nascita” e aver inizializzato alcune variabili di tipo “nascita”, ne stampi i valori componente per componente.
2. Scrivere un programma che definisca il tipo personalizzato “point” e il tipo personalizzato “circle” (sfruttando la ricorsività dei tipi derivati), inizializzi alcune variabili di questi tipi e le stampi.



# Proposte di soluzioni

## Esercizio 1:

```
PROGRAM stampa_nascita
  IMPLICIT NONE
  TYPE data
    INTEGER :: g, m, a
  END TYPE data
  TYPE nascita
    CHARACTER(132) :: luogo
    TYPE(data) :: giorno
  END TYPE nascita
```



# Proposte di soluzioni

```
TYPE (nascita) :: n
```

```
n = nascita("Milano", data(29, 2, 2004))
```

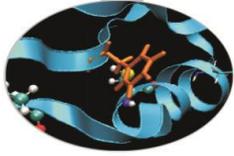
```
PRINT *, "Sono nato a ", n
```

```
WRITE (*, '(3a, 2(i2, "/"), i4)') &
```

```
"Sono nato a ", TRIM(n%luogo), &
```

```
", il giorno ", n%giorno%g, n%giorno%m, &  
n%giorno%a
```

```
END PROGRAM stampa_nascita
```



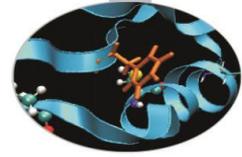
# Proposte di soluzioni

## Esercizio 2:

```
PROGRAM circleddef
  IMPLICIT NONE
  TYPE point
    REAL :: x,y
  END TYPE point

  TYPE circle
    TYPE(point) :: centre
    REAL :: radius
  END TYPE circle

  TYPE(point) :: o
  TYPE(circle) :: one
```



# Proposte di soluzioni

```
o%x = 1.0
o%y = 2.0
one%centre = o
one%radius = 0.5
WRITE (*,*) 'Centro del cerchio per componenti'
WRITE (*, '(2F6.2,/)') o%x,o%y
WRITE (*,*) 'Centro come tipo personalizzato'
WRITE (*,*) o
WRITE (*,*) 'Cerchio per componenti'
WRITE (*, '(3F6.2,/)') one
WRITE (*,*) 'Cerchio come tipo personalizzato'
WRITE (*,*) one
END PROGRAM circledef
```