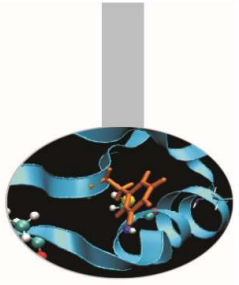
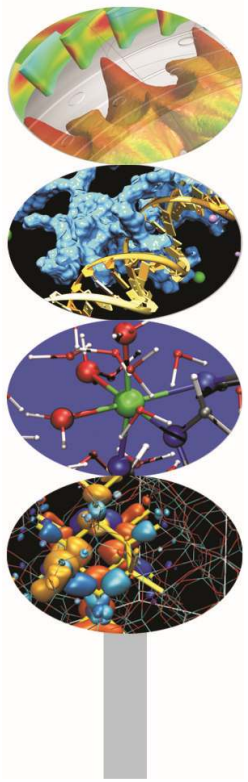
CINECA SCAI
SuperComputing Applications and Innovation



Sintassi di base



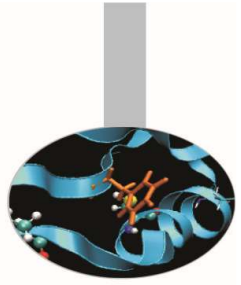
Introduction to modern Fortran

Maurizio Cremonesi, *CINECA*

Maggio 2016

CINECA

Documentazione e manualistica



Può essere facilmente reperita on-line

Language reference

Descrizione e sintassi del linguaggio:

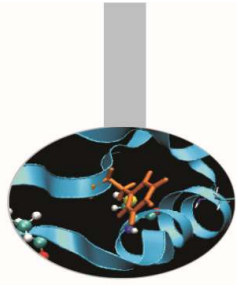
- https://gcc.gnu.org/wiki/GFortranStandards
- http://www.futa.edu.ng/materials/FORTRANTEXT.pdf

User's Guides

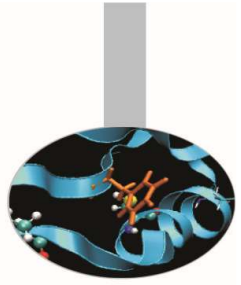
Implementazioni specifiche, opzioni del compilatore, funzioni intrinseche, codici di errore, estensioni:

https://gcc.gnu.org/onlinedocs/gfortran.pdf

Formato fisso prima del Fortran90

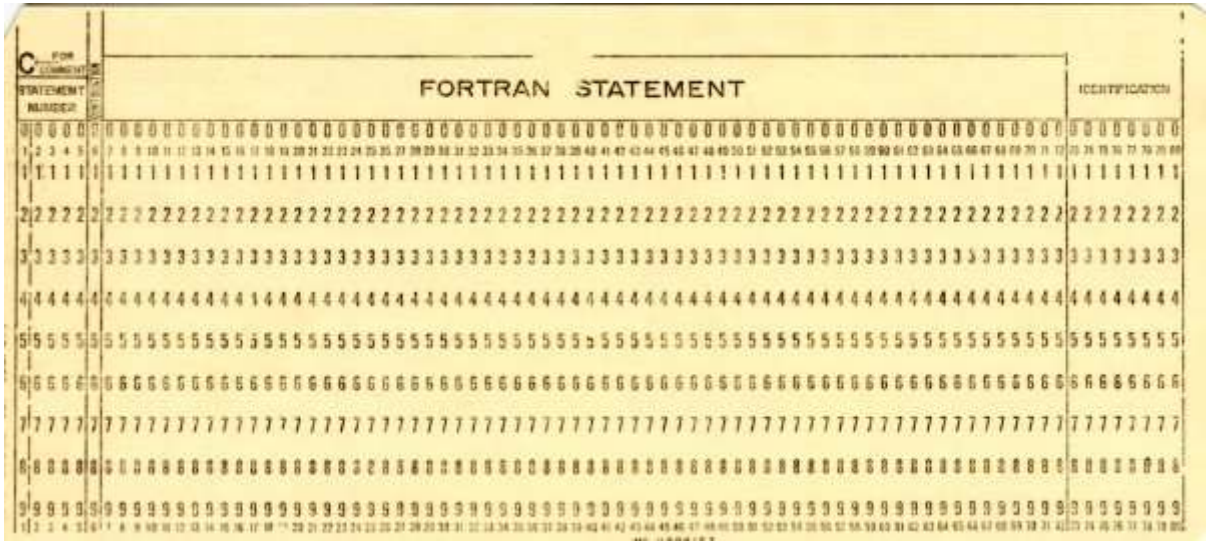


- gli spazi non sono significativi
- una riga di codice non può superare i 72 caratteri, che venivano perforati su schede nelle colonne da 1 a 72
- se la colonna 1 contiene una C (o un !), la riga è considerata un commento
- le colonne da 1 a 5 possono solo contenere un'etichetta numerica che identifica l'istruzione e che può essere usata per saltare all'istruzione stessa da un altro punto del programma
- le colonne da 7 a 72 contengono l'istruzione

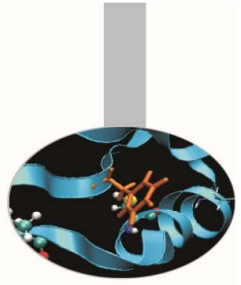


Formato fisso

- le colonne da 73 a 80 venivano usate per numerare le schede e permettere così di riordinarle nel caso fossero state accidentalmente mescolate.
- La colonna 6 è normalmente vuota ma se contiene un carattere indica che l'istruzione è il seguito dell'istruzione perforata nella scheda precedente.

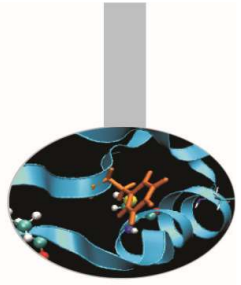


Formato fisso (esempio)



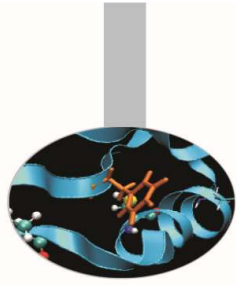
```
12345678901234567890123456789012345678901234567890

      PROGRAM CIAO
C Questo e' un commento
100   PRINT*,'Benvenuto'
      STOP
      END
```

Formato libero



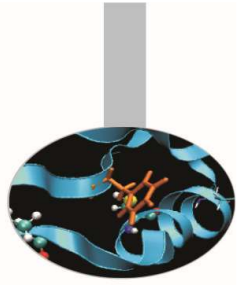
- ogni riga può contenere fino a 132 caratteri, iniziando a scrivere anche dalla colonna 1;
- gli spazi bianchi sono significativi;
- Le etichette, *il cui uso è sconsigliato*, sono identificate da un massimo di 5 caratteri numerici consecutivi scritti prima dell'istruzione, in qualsiasi colonna
- il commento è preceduto dal carattere "!", ovunque esso si trovi (anche dopo un'istruzione);
- il carattere "&" indica che una riga di codice continua nella riga successiva, fino ad un massimo di 39 righe di continuazione.

Formato libero (esempio)



```
PROGRAM CIAO
   PRINT*, "Benvenuto" ! Questo e' un commento
   STOP
END PROGRAM CIAO
```

Programmazione modulare



Ogni sezione del programma inizia e termina con un'istruzione specifica, che dipende dal tipo di sezione:

```
PROGRAM nome
    .   .   .
END PROGRAM nome
```

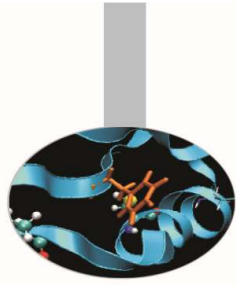
```
SUBROUTINE nome
    .   .   .
END SUBROUTINE nome
```

```
FUNCTION nome
    .   .   .
END FUNCTION nome
```

```
MODULE nome
    .   .   .
END MODULE nome
```

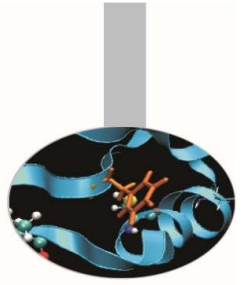
```
TYPE nome
    .   .   .
END TYPE nome
```

Struttura base



```
PROGRAM nome
   IMPLICIT NONE
   Variabili: dichiarazioni
   Istruzioni eseguibili
END PROGRAM nome
```

Nomi



I caratteri utilizzabili dal Fortran 90 sono:

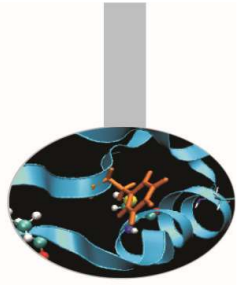
A-Z 0-9 + * / < > ; ! % ' - " & _ () $ = , .

I nomi delle entità (**variabili**, **procedure**, oggetti, strutture):

- possono avere fino a 31 caratteri, costituiti da caratteri alfanumerici (a-z, 0-9) e dal carattere underscore (_);
- devono iniziare con un carattere alfabetico;
- non possono essere parole riservate del linguaggio.

Non esiste distinzione tra maiuscole e minuscole.

Istruzioni

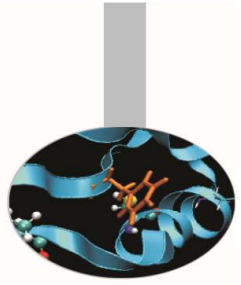


Su un'unica riga di codice possono essere scritte più istruzioni, separate da " ; " (consigliato solo in alcuni casi).

Se un'istruzione prosegue sulla riga successiva si utilizza il carattere "&" con le seguenti avvertenze:

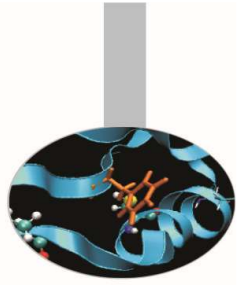
- se l'istruzione viene spezzata all'esterno di una stringa di carattere, è sufficiente inserire il carattere " & " al termine della riga precedente;
- se l'istruzione viene spezzata all'interno di una stringa di caratteri, è necessario ripetere il carattere " & " anche nella riga successiva, per evitare ambiguità di interpretazione, ovvero per chiarire che quel carattere NON è parte della stringa ma indica il concatenamento tra le righe di istruzione.

Istruzioni (esempi)



```
IF ( errore .GT. TolleranzaAmmessa ) &
     CALL GestioneMancataConvergenza(errore)

Presentazione = " Programma a supporto &
     & dell'ipotesi N. 1 sull'evoluzione &
     & delle entità osservabili elusive. &
     & Addi' 30 febbraio 3134. Versione IIA &
     & - Scritto da L'Autore Anonimo "
```

Variabili

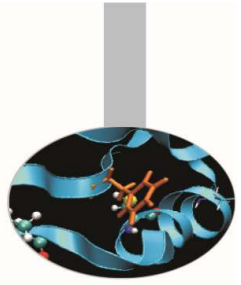


In Fortran le variabili, entità o oggetti, utilizzati nella sezione di codice, devono essere dichiarati **prima** delle istruzioni.

L'istruzione IMPLICIT NONE impone la dichiarazione esplicita di tutte le variabili che si utilizzano nel codice. Se non si usa, il compilatore assume che le variabili il cui nome inizia per [a - h] o [o - z] siano REAL, mentre quelle che iniziano con [i, j, k, l, m, n] siano INTEGER.

L'istruzione IMPLICIT NONE per questo corso dev'essere considerata *obbligatoria*; può essere preceduta solo dalle istruzioni USE e FORMAT.

Variabili



Le variabili utilizzate in Fortran possono essere riferite a cinque tipi predefiniti:

Numerici:

INTEGER = numeri interi

REAL = numeri reali (floating point)

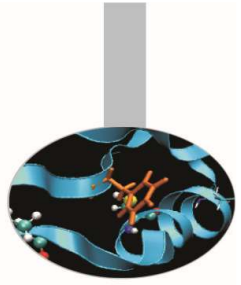
COMPLEX = numeri complessi, rappresentati da una coppia di numeri reali

Non numerici:

CHARACTER = stringhe di caratteri

LOGICAL = valori booleani (2 i valori possibili: vero / falso)

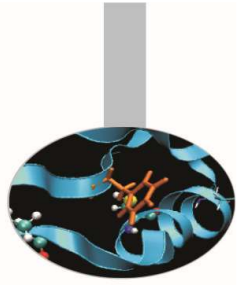
Variabili



Nel FORTRAN 77 si usa anche il tipo numerico *DOUBLE PRECISION* per i numeri reali in doppia precisione. Tale tipo è stato integrato dalla versione 90 nel tipo REAL. Perciò, nel caso di singola precisione è sufficiente dichiarare una variabile di tipo REAL, mentre nel caso di doppia precisione si deve dichiarare una variabile di tipo **REAL(8),** o utilizzare il **KIND**.

Esistono infine i tipi derivati, ovvero definiti dal programmatore come un insieme di più variabili di tipo predefinito, in modo da formare strutture di dati, che possono essere anche piuttosto complesse.

Variabili



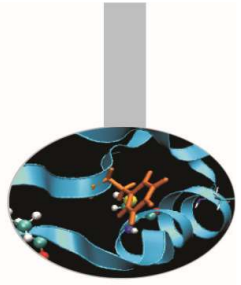
```
tipo [([KIND=]kind)] [,attributo1, attributo2, ...] &
                                :: var1 [=valore1] [,var2 [=valore2], ...]
```

tipo - può assumere i valori: INTEGER, REAL, COMPLEX, LOGICAL, CHARACTER, TYPE(nome)

kind - dipende dal tipo specificato e dall'architettura della macchina su cui si sta lavorando: è utile per specificare la precisione delle variabili.

attributo - può essere uno tra i seguenti:
PARAMETER, DIMENSION, ALLOCATABLE,
PUBLIC, PRIVATE, POINTER, TARGET, INTENT, OPTIONAL,
EXTERNAL, INTRINSIC.

Variabili



In particolare, un dato dichiarato con attributo **PARAMETER** assume un valore costante per l'intera sezione di codice in cui è dichiarato.

Valore - è il valore col quale si intende inizializzare la variabile.

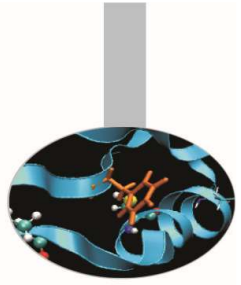
Esempio stile FORTRAN 77:

```
INTEGER statico
PARAMETER ( statico = 10 )
```

Esempio stile Fortran 90:

```
INTEGER, PARAMETER :: statico = 10, m = 1000
REAL :: a = 2.4582, b = 78.512943
```

Variabili carattere



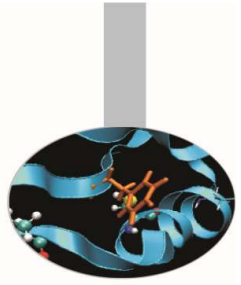
Una variabile “stringa di caratteri” è dichiarata con una lunghezza, specificabile con sintassi piuttosto articolata. Si possono anche dichiarare vettori e matrici di caratteri.

Il valore di una costante di tipo stringa di caratteri può essere definito delimitando il testo tra apici (') o virgolette (")

Se si conferisce l'attributo PARAMETER ad una stringa di caratteri, non è necessario dichiararne esplicitamente la lunghezza in quanto tale valore viene assunto in modo automatico.

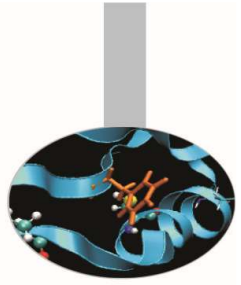
```
INTEGER, PARAMETER :: lunghezza=20
CHARACTER(LEN=lunghezza) :: nome
Nome="questa stringa"
```

Variabili carattere (esempi)



```
CHARACTER(LEN=30)  :: nome
CHARACTER(30)      :: nome
CHARACTER          :: nome*30
CHARACTER(LEN=30), DIMENSION(100) :: vettore
CHARACTER :: vettore(100)*30
CHARACTER(LEN=17), PARAMETER :: nome="contenuto stringa"
CHARACTER(*), PARAMETER :: nome="contenuto stringa"
CHARACTER, PARAMETER    :: nome="contenuto stringa"
```

Variabili carattere



In questo corso si userà di preferenza la sintassi:

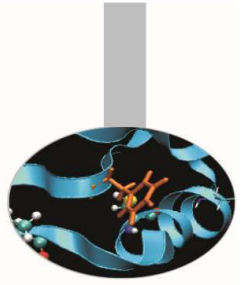
```
CHARACTER(lunghezza) :: nome
```

o la variante

```
CHARACTER(lunghezza), DIMENSION(n,m) :: nome
```

nel caso di matrici di stringhe di caratteri

Variabili carattere



Anche la sintassi per la manipolazione di stringhe di caratteri è peculiare:

Estrazione di una sotto-stringa:

```
sottostringa = stringa (inizio:fine)
```

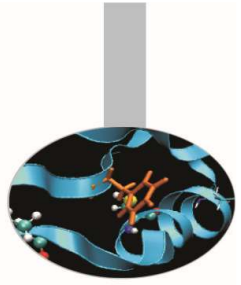
Concatenamento di due stringhe:

```
stringa = prima(inizio:fine) // seconda(inizio:fine)
```

Funzione intrinseca che ritorna la lunghezza effettiva del testo:

```
i = LEN_TRIM (stringa)
```

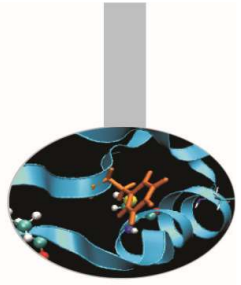
Sintassi di base



Esempio:

```
PROGRAM semplice2
   IMPLICIT NONE
   REAL :: n, m
   INTEGER, PARAMETER :: LS=40
   CHARACTER(LS) :: nome

   PRINT*,"Qual'e' il tuo nome?"
   READ*,nome

   n = 2; m = 3

   PRINT*,"Il tuo nome e' ",nome
   PRINT*," e ",n," + ",m," = ",(n + m)

   STOP
END PROGRAM semplice2
```

Sintassi di base



Esempio:

```
PROGRAM articolato1
   IMPLICIT NONE
   REAL :: n, m, s
   INTEGER, PARAMETER :: LS=80
   CHARACTER(LS) :: nome, testo

   PRINT*,"Qual'e' il tuo nome?"
   READ*,nome

   testo = "Il tuo nome e' "// &
      & TRIM(nome)//" e "

   n = 2; m = 3
   CALL SOMMA(s,n,m)

   PRINT*,TRIM(testo),n, &
      & " + ",m," = ",s

   STOP
END PROGRAM articolato1
```

```
SUBROUTINE SOMMA(r,a,b)
   IMPLICIT NONE
   REAL :: r, a, b

   r = a + b

   RETURN
END SUBROUTINE
```

Esercizio:
modificare l'esempio aggiungendo lettura e stampa del cognome aggiungere e utilizzare la subroutine che calcola la moltiplicazione