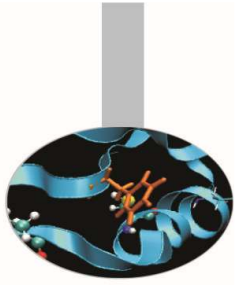
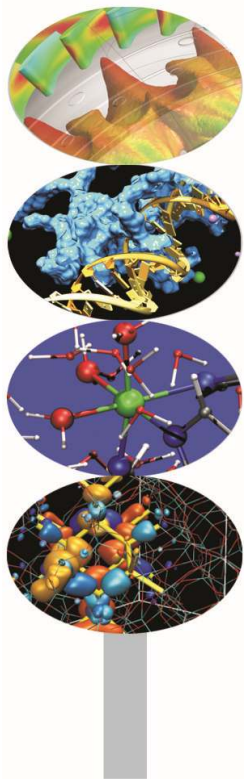
CINECA SCAI
SuperComputing Applications and Innovation





Procedure intrinseche

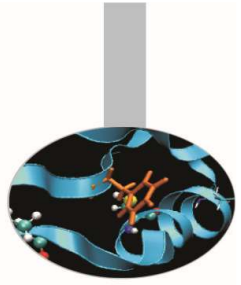
Introduction to modern Fortran

Maurizio Cremonesi, *CINECA*

Maggio 2017

CINECA

Procedure intrinseche

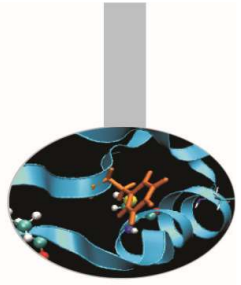


Allo scopo di rendere più chiaro il programma, è buona norma, laddove vengono utilizzate procedure intrinseche, segnalarle con la dichiarazione `INTRINSIC`.

La parola riservata INTRINSIC è seguita dalla lista dei nomi delle procedure intrinseche utilizzate.

Nel caso vengano usate procedure intrinseche esclusive del compilatore, non riconosciute dallo standard, la presenza della dichiarazione INTRINSIC permetterà di evidenziare l'eventuale non disponibilità della funzione.

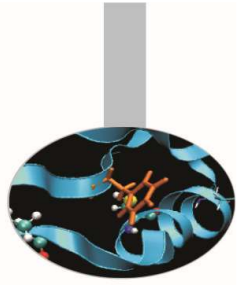
Procedure intrinseche



Esempio:

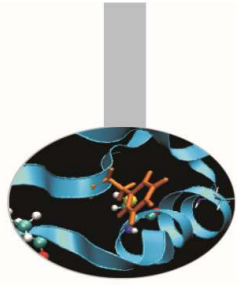
```
   IMPLICIT NONE
   REAL(8) :: x
   INTRINSIC :: SQRT
!
   PRINT*,"Trasmetti un numero"
   READ*,x
   PRINT*,"La radice quadrata di ",x,&
                  " e' ",SQRT(x)
```

Procedure intrinseche



`CALL DATE_AND_TIME([DATE] [,TIME] [,ZONE] [VALUES])`	ritorna data e ora
`CALL RANDOM_NUMBER(HARVEST)`	ritorna una sequenza casuale nell'intervallo [0,1).
`CALL RANDOM_SEED([SIZE] [,PUT] [,GET])`	inizializza il generatore di numeri casuali
`CALL SYSTEM_CLOCK([COUNT] [,COUNT_RATE] [COUNT_MAX])`	ritorna il numero di cicli dell'orologio di sistema
`CALL CPU_TIME(t1)`	ritorna il tempo di CPU in secondi (Fortran 95)

Procedure intrinseche



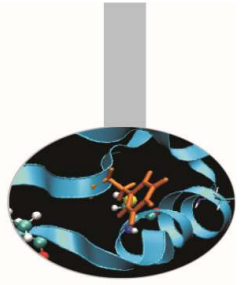
DATE_AND_TIME

Ritorna data e ora dell'istante in cui viene richiamata.

I tempi sono ritornati sia come stringa di caratteri, sia come numeri.

DATE_AND_TIME è una procedura abbastanza complessa, non adatta a misurare tempi di calcolo piccoli; a questo scopo è meglio usare la SYSTEM_CLOCK.

Procedure intrinseche



Sintassi :

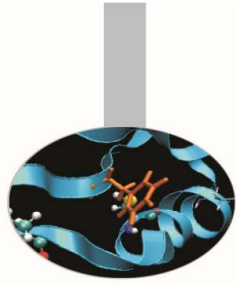
```
CHARACTER :: data*8, ora*10, zona*5
CALL DATE_AND_TIME(DATE=data,TIME=ora &
                   ZONE=zona)
```

`data`: giorno nella forma stringa di caratteri "YYYYMMGG"

`ora`: ora nella forma stringa di caratteri "HHMMSS.mmm"

`zona`: differenza rispetto all'ora di Greenwich nella forma stringa di caratteri "+HHMM"

Procedure intrinseche



Sintassi :

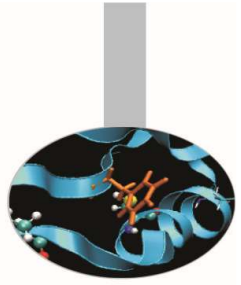
```
INTEGER, DIMENSION(8) :: tempo
CALL DATE_AND_TIME(VALUES=tempo)
```

`tempo(1:3)` : anno, mese, giorno

`tempo(4)` : differenza in minuti rispetto all'ora di Greenwich

`tempo(5:8)` : ora, minuti, secondi, millisecondi

Procedure intrinseche

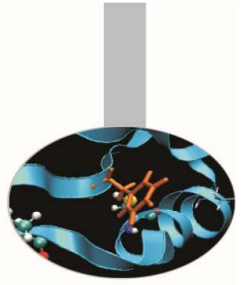


RANDOM_NUMBER

Questa subroutine ritorna una sequenza casuale di numeri, di distribuzione uniforme nell'intervallo [0,1).

La procedura RANDOM_NUMBER è elementale, perciò il suo valore prende la forma dell'oggetto passato in argomento, ovvero se viene passato uno scalare, il risultato è un singolo valore scalare, se vengono passati un vettore o una matrice il risultato è vettore o matrice.

Procedure intrinseche



La sequenza ritornata da RANDOM_NUMBER è pseudo-casuale, ovvero i valori sono calcolati con un algoritmo.

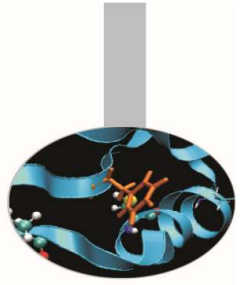
In particolare la sequenza dei valori dipende da un parametro chiamato seme (seed).

Se da un lato questo può rappresentare un problema per certe applicazioni, ha invece il pregio della ripetibilità, garantita sotto precise condizioni.

Algoritmi che si basano sul calcolo di numeri casuali possono richiedere l'utilizzo di librerie più sofisticate, ad esempio PRNG:

http://sprng.cs.fsu.edu/sprng.html

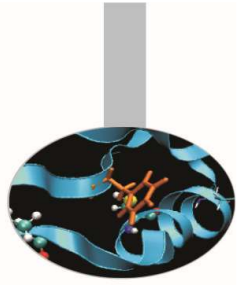
Procedure intrinseche



Esempio:

```
REAL :: casov
CALL RANDOM_NUMBER(casov)
PRINT*,casov
```

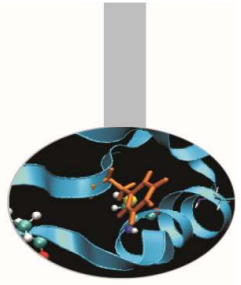
Procedure intrinseche



Esercizio 5:

Realizzare un programma che, in un ciclo iterativo indefinito, chieda di indovinare un numero da 1 a 10, da confrontare quindi con il valore ritornato da RANDOM_NUMBER (opportunamente rinormalizzato).

Procedure intrinseche

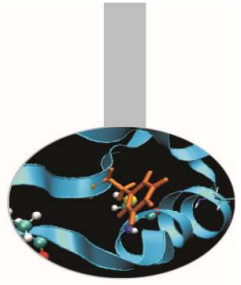


RANDOM_SEED

Questa subroutine ritorna informazioni sul parametro *seed*, che determina la sequenza dei valori prodotti dalla RANDOM_NUMBER.

Essa permette di riassegnare il parametro ogni volta che si vuole ripetere la sequenza.

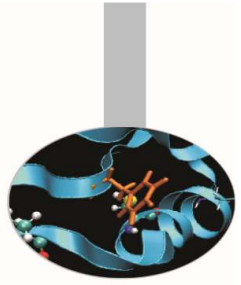
Procedure intrinseche



Uso della RANDOM_SEED:

1. si richiede di avere il valore di *SIZE*, necessario perché il seme è un vettore, la cui lunghezza dipende dal compilatore e dal calcolatore utilizzati
2. sulla base del valore ritornato da *SIZE* si dimensiona opportunamente il vettore che contiene il seme
3. con *GET* si memorizza il valore del seme in un apposito vettore
4. ogni volta che è necessario la sequenza venga ripetuta, si ristabilisce il valore del seme con *PUT*

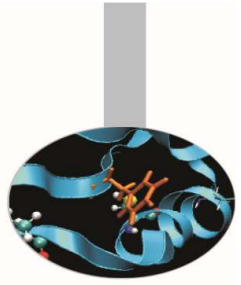
Procedure intrinseche



Esempio (RANDOM_SEED ammette 1 solo argomento alla volta):

```
   INTEGER :: s
   INTEGER, DIMENSION(LEN) :: mioseme, seme
   CALL RANDOM_SEED(SIZE=s)
!     s: output – lunghezza del vettore
   CALL RANDOM_SEED(GET=seme)
!     seme: output - vettore contenente il seme usato
                     dal generatore
   CALL RANDOM_SEED(PUT=mioseme)
!     mioseme: input - vettore contenente un seme
                    alternativo, o quello originale
```

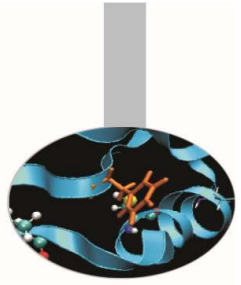
Procedure intrinseche



SYSTEM_CLOCK

In ogni calcolatore le operazioni sono sincronizzate con un orologio i cui battiti sono memorizzati in un apposito registro e da lì possono essere letti utilizzando questa subroutine.
L'operazione impegna poco il calcolatore, quindi la *SYSTEM_CLOCK* è adatta a misurare il tempo di calcolo di porzioni di codice anche abbastanza brevi.

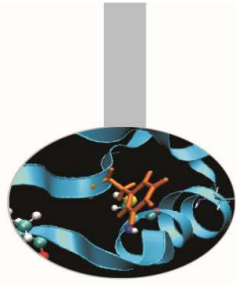
Procedure intrinseche



Esempio:

```
INTEGER :: cicli0, cicli1
CALL SYSTEM_CLOCK(COUNT=cicli0)
        .   istruzioni .
CALL SYSTEM_CLOCK(COUNT=cicli1)
PRINT*,"Battiti misurati: ", &
                        (cicli1-cicli0)
```

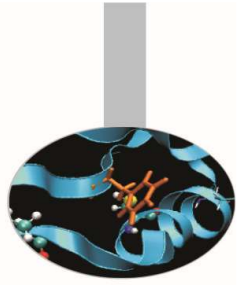
Procedure intrinseche



Una complicazione da tener presente è che il registro ha uno spazio limitato per conservare il numero di battiti dell'orologio, perciò superato un massimo, il valore si riazzera:

```
CALL SYSTEM_CLOCK(COUNT=cicli0, &
                  COUNT_MAX=cicli_massimi)
      .   .   .
CALL SYSTEM_CLOCK(COUNT=cicli1)
IF (cicli1 < cicli0) cicli1 = &
                    cicli1 + cicli_massimi
```

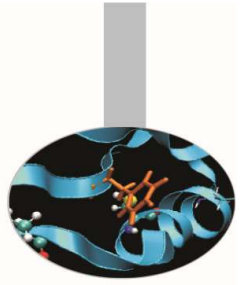
Procedure intrinseche



Se è richiesta una misura temporale più familiare, con l'argomento *COUNT_RATE* è possibile sapere quanti battiti di orologio vengono fatti ogni secondo e calcolare quindi il tempo in secondi:

```
CALL SYSTEM_CLOCK(&
                  COUNT_RATE=cicli_al_secondo)
```

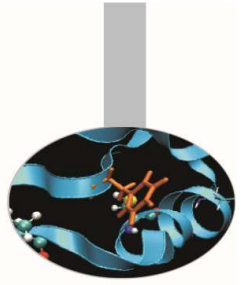
Procedure intrinseche



Esempio:

```
CALL SYSTEM_CLOCK(COUNT=cicli0, &
         COUNT_RATE=cicli_al_secondo, &
         COUNT_MAX=cicli_massimi)
       .    .    .
CALL SYSTEM_CLOCK(COUNT=cicli1)
If ( cicli1 < cicli0 ) &
           cicli1 = cicli1 + cicli_massimi
cicli = (cicli1-cicli0)
frequenza = cicli_al_secondo
PRINT*,"Secondi richiesti dal codice: ", &
          DBLE(cicli)/DBLE(frequenza)
```

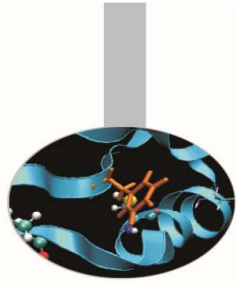
Procedure intrinseche



Esercizio 6:

Nel programma dell'esercizio 5 usare RANDOM_SEED per modificare il seme con il valore ritornato da SYSTEM_CLOCK. Può essere simpatico usare preventivamente DATE_AND_TIME per presentare data e ora di esecuzione del programma.

Procedure intrinseche

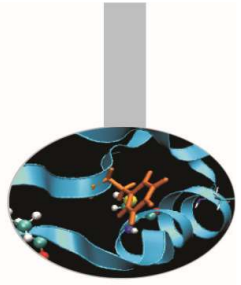


CPU_TIME

Questa subroutine, introdotta dal Fortran 95, ritorna il tempo di CPU in secondi:

```
        REAL: cput, cput0, cput1
           .   .   .
        CALL CPU_TIME(cput0)
           .   .   .
  !     Istruzioni da misurare
           .   .   .
        CALL CPU_TIME(cput1)
        cput = cput1 - cput0
```

Procedure intrinseche (Fortran 2003)

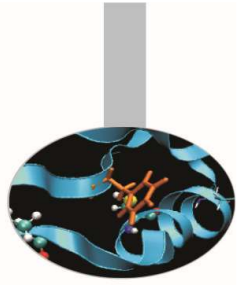


Il Fortran 2003 introduce alcune procedure e costanti utili per standardizzare i rapporti con l'ambiente operativo.

Col Fortran 2003 compare il concetto di modulo intrinseco.

Uno di questi moduli si chiama *ISO_FORTRAN_ENV* e contiene tra l'altro alcune funzioni di cui effettivamente si sentiva la mancanza in Fortran.

Procedure intrinseche (Fortran 2003)



```
COMMAND_ARGUMENT_COUNT()
```

ritorna il numero dei parametri passati al programma

```
CALL GET_COMMAND(comando, lunghezza, stato)
```

ritorna la riga di comando usata per eseguire il programma

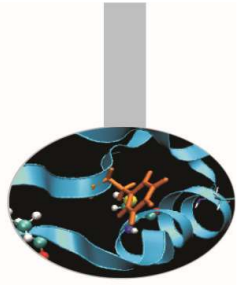
```
CALL GET_COMMAND_ARGUMENT(number, &
                          value, length, status)
```

ritorna il parametro dato a riga di comando, o il nome dell'eseguibile

```
CALL GET_ENVIRONMENT_VARIABLE(name, value,&
                              length, status)
```

ritorna il valore di un parametro di sistema

Procedure intrinseche (Fortran 2003)



Esempio *parametri:*

```
   quanti = COMMAND_ARGUMENT_COUNT()
   IF ( quanti < 1 ) THEN
      PRINT*,"Modalita' di esecuzione: &
                    & fattoriale <n>"
      STOP
   END IF
!  . . . altrimenti si stampa la riga di
!          comando utilizzata
   CALL GET_COMMAND(rg, LENGTH=lun, STATUS=st)
   IF ( st <= 0 ) THEN
      PRINT*,"Hai trasmesso: ",rg(1:lun)
   END IF
```