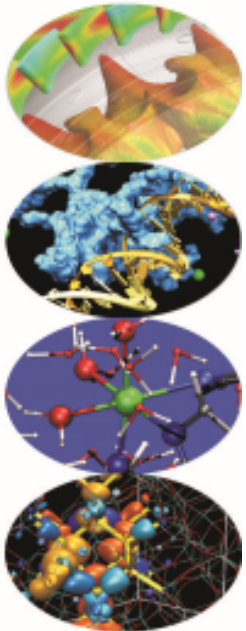


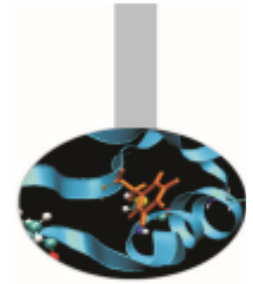
# Precisione dei dati

## *Introduction to Fortran 90*

Maurizio Cremonesi, *CINECA*

Marzo 2015

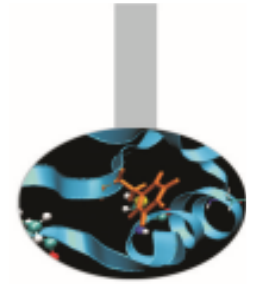




# Introduzione

In Fortran abbiamo a disposizione **cinque tipi predefiniti** per dichiarare le variabili: `INTEGER`, `REAL`, `COMPLEX`, `CHARACTER`, `LOGICAL`.

Associati ai **tipi** ci sono le **specie**, definite tramite l'attributo `KIND`, che determinano anche la quantità di memoria da riservare.



# Funzione KIND()

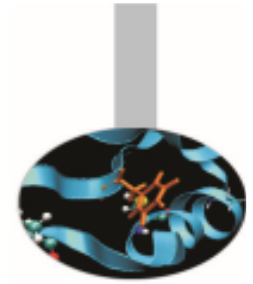
La funzione intrinseca `KIND()` ritorna un intero, dipendente dal sistema, corrispondente alla precisione richiesta.

Tale valore può essere assegnato, in fase di dichiarazione, a un `PARAMETER` oppure direttamente all'attributo `KIND`.

## Esempio:

```
INTEGER, PARAMETER :: tipo_doppio = KIND(1.0D0)
REAL(KIND=tipo_doppio) :: a,b

REAL(tipo_doppio) :: c,d
REAL(KIND=KIND(1.0D0)) :: e,f
```



# Funzione KIND()

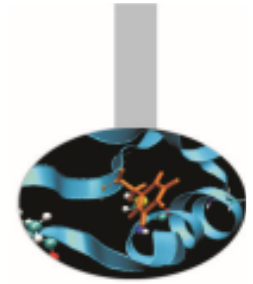
La funzione intrinseca `KIND()` può anche essere utilizzata per **verificare il kind** di una variabile.

## Esempio:

```
INTEGER, PARAMETER :: tipo_doppio = KIND(1.0D0)
REAL(KIND=tipo_doppio) :: a,b,c
WRITE(*,*) KIND(a), KIND(b), KIND(c)
```

## Esempio portabilità codice:

```
INTEGER, PARAMETER :: tipo_doppio = KIND(1.0D0)
REAL(tipo_doppio) :: a,b,c !codice portabile
REAL *8 :: d,e,f ! Sintassi f77 non assicura
! la portabilità.
```



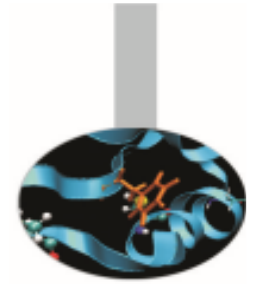
# Funzioni e portabilità

Qualora si intenda lavorare con valori reali, complessi e interi che richiedano una determinata precisione, indipendentemente dal sistema di calcolo utilizzato, il Fortran 90 mette a disposizione del programmatore due funzioni specifiche:

`SELECTED_REAL_KIND()` (reali e complessi)

`SELECTED_INT_KIND()` (interi)

Tramite queste funzioni è assicurata la portabilità del codice da una piattaforma ad un'altra, mantenendo la precisione richiesta.

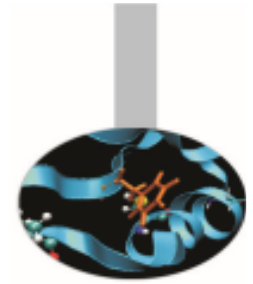


# Funzioni e portabilità

La funzione `SELECTED_REAL_KIND(cifre, esp)` si applica a reali e complessi. Essa riceve in argomento due interi che indicano rispettivamente il numero di cifre decimali ed il range dell'esponente e ritorna il kind che permette di rappresentare un numero con la precisione specificata.

## Esempio:

```
INTEGER, PARAMETER :: i10 = &  
                        & SELECTED_REAL_KIND (10,200)  
REAL (KIND = i10) :: a, b, c ! Le variabili a, b, c  
hanno almeno 10 cifre decimali di precisione e  
l'esponente nel range da -200 a +200.
```

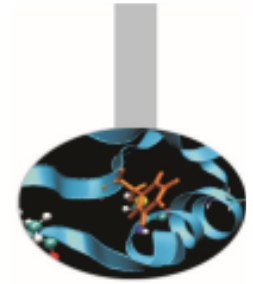


# Funzioni e portabilità

Si deve notare tuttavia che il valore di kind ritornato è quello predefinito per il sistema, perciò usualmente sarà la precisione semplice, doppia, estesa ed eventualmente la quadrupla.

## Esempio :

compilatore intel	kind ritornato	precisione
<code>SELECTED_REAL_KIND(3,1)</code>	4	singola
<code>SELECTED_REAL_KIND(8,100)</code>	8	doppia
<code>SELECTED_REAL_KIND(28,100)</code>	-1	non rapp.
compilatore salford	kind ritornato	precisione
<code>SELECTED_REAL_KIND(3,1)</code>	1	singola
<code>SELECTED_REAL_KIND(8,100)</code>	2	doppia
<code>SELECTED_REAL_KIND(28,100)</code>	-1	non rapp.



# Funzioni e portabilità

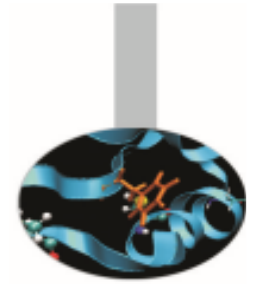
E' bene precisare che l'entità che specifica il `KIND` nelle dichiarazioni deve essere un `PARAMETER`. Essa può venire usata direttamente per specificare il `KIND` delle costanti che compaiono nel programma.

## Esempio:

```
INTEGER, PARAMETER :: poca=SELECTED_REAL_KIND(3,1)
INTEGER, PARAMETER :: molta=SELECTED_REAL_KIND(8,1)
REAL(KIND=poca)    :: a
REAL(KIND=molta)   :: da
```

```
a = 1.0_poca
da = 1.0_molta
```





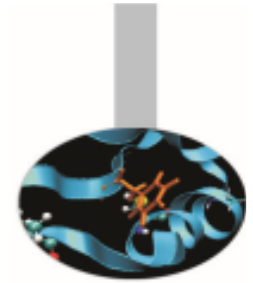
# Funzioni e portabilità

La funzione `SELECTED_INT_KIND(esp)` si applica alle variabili intere. Essa riceve in argomento un intero che indica il range dell'esponente e ritorna il kind che permette di rappresentare un numero con la precisione specificata.

## Esempio:

```
INTEGER, PARAMETER :: i8 = SELECTED_INT_KIND ( 8 )  
INTEGER (KIND = i8) :: ia, ib, ic
```

Le variabili `ia`, `ib`, `ic` avranno valori compresi fra  $-10^8$  e  $+10^8$



# Funzioni Intrinseche

SELECTED_INT_KIND (R)	valore dell'attributo KIND del tipo intero che soddisfa l'esponente intero R. -1 se non esiste
SELECTED_REAL_KIND (P, R)	valore dell'attributo KIND del tipo reale che soddisfa la precisione decimale P e l'esponente intero R. -1 se non esiste
KIND (X)	valore del parametro KIND per l'entità X
LOGICAL (L [, KIND])	conversione tra tipi logici

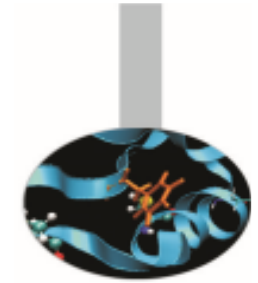
```
INTEGER, PARAMETER :: pr_r=SELECTED_REAL_KIND(8,100)
```

```
INTEGER, PARAMETER :: pr_i=SELECTED_INT_KIND(8)
```

```
REAL(KIND=pr_r) :: a
```

```
REAL(KIND=pr_i) :: b
```

```
WRITE(*,*) kind(a), kind(b)
```



# Funzioni Intrinseche

BIT_SIZE (I)	numero massimo di bit per il tipo intero rappresentato da I
CEILING (A)	il più piccolo intero non inferiore al reale A
FLOOR (A)	il più grande intero non superiore al reale A
MOD (A, P)	resto della divisione. $A - \text{INT}(A/P) * P$
MODULO (A, P)	modulo per argomenti entrambi reali o interi. $A - \text{FLOOR}(A/P)*P$

## Esempi:

```
INTEGER :: i = -8, j = 5
```

```
REAL :: a = 2.16
```

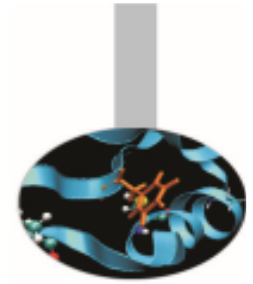
```
WRITE (*,*) 'Bit size      : ', BIT_SIZE (j)      ! 32
```

```
WRITE (*,*) 'Ceiling       : ', CEILING (a)       ! 3
```

```
WRITE (*,*) 'Floor         : ', FLOOR (a)         ! 2
```

```
WRITE (*,*) 'Mod           : ', MOD (i,j)          ! -3
```

```
WRITE (*,*) 'Modulo        : ', MODULO (i,j)       ! 2
```

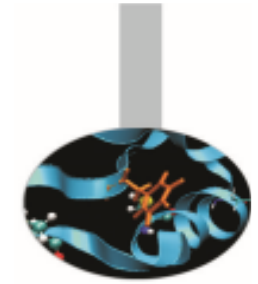


# Funzioni Intrinseche

Le funzioni intrinseche delle slides successive sono basate sul seguente modello di rappresentazione dei reali:

$$x = sb^e \sum_{k=1}^p f_k b^{-k}$$

- x:** valore della variabile reale;
- s:** segno (+/-1);
- b:** base il cui valore è un intero > 1 praticamente sempre uguale a 2;
- e:** intero pari al valore dell'esponente necessario a rappresentare il valore **x** nella base caratteristica dell'architettura della macchina;
- p:** numero di bit della mantissa per la rappresentazione in virgola mobile;
- f<sub>k</sub>:** valore del k-esimo digit. Tale valore è compreso tra  $0 \leq f_k < b$ .

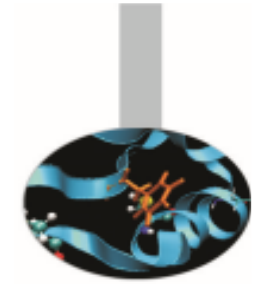


# Funzioni Intrinseche

EXPONENT (X)	numero di bit dell'esponente nella rappresentazione del reale X ; Corrisponde al valore dell'esponente "e".
FRACTION (X)	parte frazionaria del reale X. Il risultato di questa funzione è dato da $X \cdot b^{-e}$ e rappresenta il valore della sommatoria nella rappresentazione sopra riportata.
NEAREST (X, S)	il più vicino numero diverso da X nella direzione del segno di S
SPACING (X)	spacing assoluto della rappresentazione del reale X; corrisponde al valore risultante da : $b^{e-p}$ .

```

INTEGER, PARAMETER :: ik = SELECTED_REAL_KIND(10)
REAL(ik) :: pi=3.14159265358979323846_ik, in = 1.0_ik
WRITE (*,*) `Exponent      : ', EXPONENT (pi)      ! 2
WRITE (*,*) `Fraction      : ', FRACTION (pi)      ! 0.7853..
WRITE (*,*) `Nearest      : ', NEAREST (pi,in)    ! 3.1415..
WRITE (*,*) `Spacing      : ', RRSPACING (pi)     ! 4.44E-16
  
```

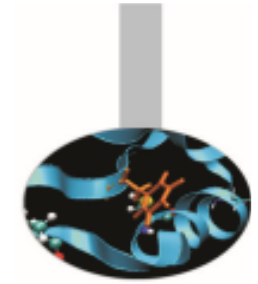


# Funzioni Intrinseche

SCALE (X, I)	$X * b^I$
SET_EXPONENT (X, I)	numero reale la cui parte frazionaria è quella di X ed il cui esponente è I
RRSPACING (X)	reciproco dello spacing relativo nella rappresentazione del reale X

```

INTEGER, PARAMETER :: ik = SELECTED_REAL_KIND(10)
REAL(ik) :: pi=3.14159265358979323846_ik, in = 1.0_ik
INTEGR :: n=3
WRITE (*,*) `Scale          : ', SCALE (pi,n)           ! 25.13
WRITE (*,*) `Setexponent   : ', SET_EXPONENT (pi,n)     ! 6.28
WRITE (*,*) `RRspacing     : ', RRSPACING (pi)          ! 7.07E+15
  
```

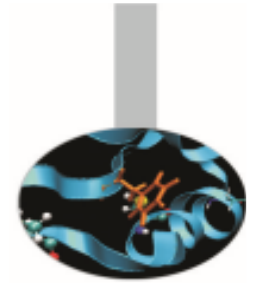


# Funzioni Intrinseche

DIGITS (X)	numero di bit significativi per la rappresentazione del numero intero o reale X, tolti il segno e l'esponente
EPSILON (X)	numero più piccolo tale che, aggiunto a 1, gli cambia valore. Il risultato è dello stesso tipo del reale X ed è dato da $b^{1-p}$ .
HUGE (X)	il più grande numero positivo rappresentabile, dello stesso tipo del reale X
MAXEXPONENT (X)	il più grande esponente del 2 per i numeri rappresentabili, dello stesso tipo del reale X

```

INTEGER, PARAMETER :: ik = SELECTED_REAL_KIND(10)
REAL(ik) :: pi=3.14159265358979323846_ik, in = 1.0_ik
WRITE (*,*) `Digits      : ', DIGITS (pi)      ! 53
WRITE (*,*) `Epsilon    : ', EPSILON (pi)     ! 2.22E-16
WRITE (*,*) `Huge       : ', HUGE (pi)        ! 1.79E308
WRITE (*,*) `Maxexponent : ', MAXESPONENT (pi) ! 1024
  
```



# Funzioni Intrinseche

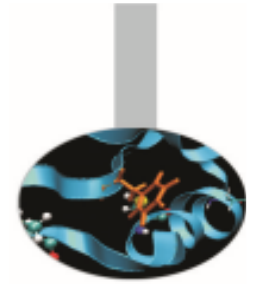
MINEXPONENT (X)	il più grande esponente negativo del 2 per i numeri di tipo X
PRECISION (X)	precisione decimale per i reali dello stesso tipo di X
RADIX (X)	base nella rappresentazione dei numeri dello stesso tipo di X ;
RANGE (X)	esponente decimale massimo per i numeri, interi, reali, complessi dello stesso tipo di X
TINY (X)	il più piccolo numero positivo reale dello stesso tipo di X

```

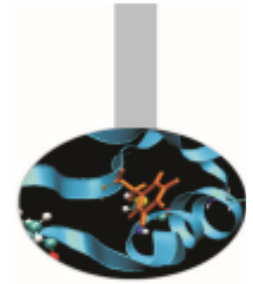
INTEGER, PARAMETER :: ik = SELECTED_REAL_KIND(10)
REAL(ik) :: pi=3.14159265358979323846_ik, in = 1.0_ik
WRITE (*,*) `Minexponent   : ', MINEXPONENT (pi) ! -1021
WRITE (*,*) `Precision     : ', PRECISION (pi)   ! 15
WRITE (*,*) `Radix        : ', RADIX (pi)       ! 2
WRITE (*,*) `Range        : ', RANGE (pi)       ! 307
WRITE (*,*) `Tiny         : ', RANGE (pi)       ! 2.22E-308
  
```



# Esempi



1. Definizione e valori di RRSPACING e SPACING: [Spacing.f90](#)
2. Valori di kind per le variabili intere del sistema: [Kind\\_int.f90](#)
3. Verifica delle differenze tra le funzioni MOD e MODULO:  
[Mod\\_Modulo.f90](#)



## Esercizi (facoltativi)

1. Scrivere un programma che ritorna i valori delle funzioni intrinseche DIGITS(), HUGE(), RANGE() per gli interi dei KIND disponibili. Si suggerisce di usare i kind presentati dall'esempio `Kind_int.f90`
2. Fare la stessa cosa con le funzioni DIGITS(), HUGE(), TINY(), RANGE() applicate ai reali.