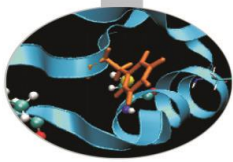
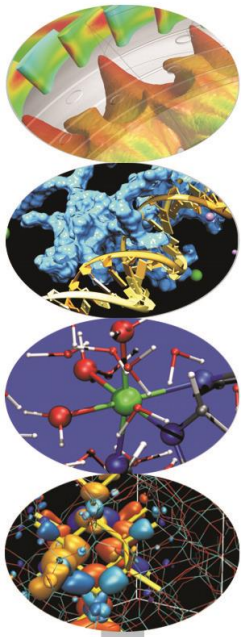
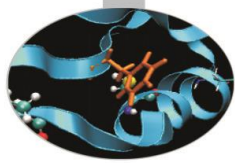


Input Output

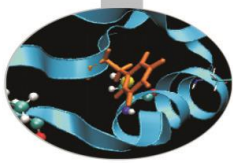
esercitazioni



Esercizi



1. Scrivere un programma contenente un ciclo DO che legge numeri reali da un file esterno, salta i numeri negativi, si interrompe se legge zero, somma la radice quadrata dei numeri positivi (usare EXIT e CYCLE) e scrive il risultato su file. Confrontare i diversi risultati che si ottengono usando il parametro DELIM (con valori QUOTE/APOSTROPHE/NONE) nella OPEN.
2. Scrivere un programma per salvare su file ad accesso diretto non formattato una matrice riga per riga e quindi rileggerla in ordine inverso.



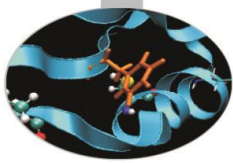
Esercizi

3. Le diverse modalità di accesso alle unità esterne e di trasferimento dei dati hanno un forte impatto non solo sul risultato finale, ma anche sulle prestazioni del codice. Estendere la traccia relativa all'esercizio 3 contenente l'accesso sequenziale formattato al caso di accesso diretto non-formattato, diretto formattato, sequenziale non-formattato. Verificare le performance. Per misurare il tempo solare richiesto da una porzione di programma può risultare utile la funzione intrinseca DATE_AND_TIME di cui si riporta un'esempio di utilizzo:

```
integer, dimension(1:8) :: Tc

call date_and_time(values=Tc)
secc = Tc(5)*3600.0 + Tc(6)*60.0 +
     &       Tc(7)*1.0 +  Tc(8)*0.001
```

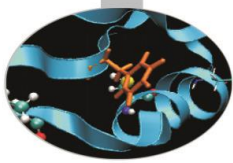
Soluzioni



Esercizio 1/0

```
PROGRAM square_sum
  IMPLICIT NONE
  REAL :: sq_sum, number
  OPEN (11, FILE = 'square.dat')
  OPEN (22, FILE='square.ris', STATUS="REPLACE", DELIM="quote")
  sq_sum = 0.0
  DO
    READ (11, '(F4.2)') number
    WRITE (22,*) 'number = ', number
    IF (number == 0.0) EXIT
IF (number <  0.0) CYCLE
    sq_sum = sq_sum + SQRT(number)
    WRITE (22,*) 'square_sum =', sq_sum
  END DO
```

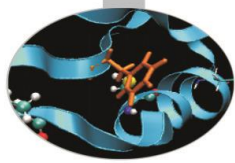
Soluzioni



Esercizio 1/1

```
WRITE (22,*) ''
  WRITE (22,*) 'The solution is:'
  WRITE (22,*) sq_sum
  STOP
END PROGRAM square_sum
```

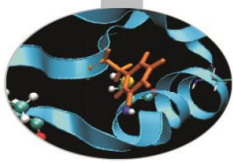
Soluzioni



Esercizio 2/0

```
program salva_diretto
        implicit none
        integer i, j, k, lunrec, n
        integer mat(5,10)
        real number
        do i = 1, 10
           do j = 1, 5
              mat(j,i) = (j-1)*10 + i
           enddo
        enddo
```

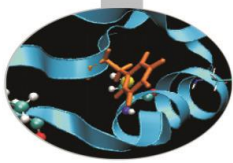
Soluzioni



Esercizio 2/1

```
 lunrec=10*4
OPEN(UNIT = 11, FILE = 'salva.dun', RECL = lunrec,
& ACCESS='direct', FORM = 'UNFORMATTED', &
STATUS="REPLACE")
          do j = 1, 5
             WRITE(11,REC=j,IOSTAT=n)
(mat(j,i),i=1,10)
             if (n.NE.0) then
                print*,' Errore scrittura record ',j
             end if
          enddo
          CLOSE(UNIT = 11)
```

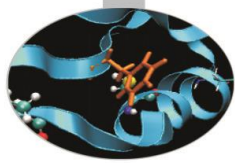
Soluzioni



Esercizio 2/2

```
! Si azzera la matrice per sicurezza
mat(:,:) = 0
OPEN(UNIT = 22, FILE = 'salva.dun', RECL = lunrec,
&  ACCESS='direct', FORM = 'UNFORMATTED', STATUS="OLD")
      do j = 1, 5
         READ(22,REC=(6-j),IOSTAT=n)  (mat(j,i),i=1,10)
         if (n.NE.0) then
            print*,' Errore lettura record ',j
         end if
      enddo
CLOSE(UNIT = 22)
```

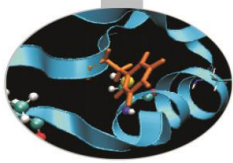
Soluzioni



Esercizio 2/3

```
do j = 1, 5
   write(*,100) (mat(j,i),i=1,10)
enddo
100        format(1x,10(i6.2))
stop
end
```

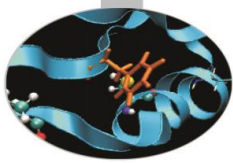
Soluzioni



Esercizio 3/0 (Traccia)

```
Program Test_1
  integer, parameter :: l1 = 200, l2 = 200
  real, dimension(l1,l2) :: B
  real, dimension(l2,l1) :: BT
  real, dimension(l1,l1) :: C
  real, dimension(l2,l2) :: A
  real, dimension(l2,l1) :: P1
  integer :: i, j, l
  real :: COEFF1, COEFF2
  integer, dimension(1:8) :: T0,Tc,Tf
  real :: sec0, secf,secc
  character(len=8) :: nomefile
```

Soluzioni



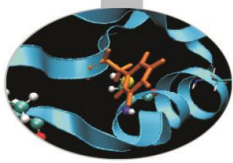
Esercizio 3/1

```
! Richiesta del nome del file di output e relativa
!apertura dello stesso

  write(unit=*, fmt=*) " Nome del file di output "
  read(unit=*, fmt=*) nomefile
  open(unit=10, file=nomefile, status="replace", &
  form="formatted", action="write",delim="quote")

! Valutazione del tempo iniziale
  call date_and_time(values=T0)
  sec0 = T0(5)*3600.0 + T0(6)*60.0 + T0(7)*1.0 + &
  T0(8)*0.001
```

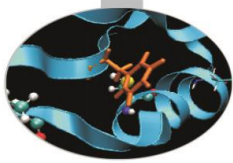
Soluzioni



Esercizio 3/2

```
! Generazione delle matrici e dei parametri con
numeri !casuali
 call random_number(COEFF1)
 call random_number(COEFF2)
 call random_number(B)
 call random_number(C)
 ! Calcolo della matrice [B] trasposta
 BT=transpose(B)
 ! Prodotto matriciale tra [BT] e [C]
 P1 = matmul(BT,C)
 ! Prodotto matriciale tra [P1] e [B]
 A =(matmul(P1,B))*(COEFF1*COEFF2)
```

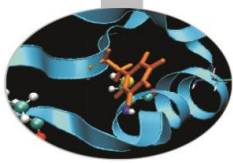
Soluzioni



Esercizio 3/3

```
! Valutazione del tempo di calcolo
 call date_and_time(values=Tc)
 secc = Tc(5)*3600.0 + Tc(6)*60.0 + Tc(7)*1.0 + &
        Tc(8)*0.001
 ! Stampa della matrice [B]
 write(unit=10, fmt="(a)") " Stampa della matrice
[B] "
 write(unit=10, fmt="(a)") " "
 do i = 1, l1
    write(unit=10, fmt="(5 (f10.5,2x))") B(i,:)
 end do ! i
 write(unit=10, fmt="(a)") " "
```

Soluzioni



Esercizio 3/4

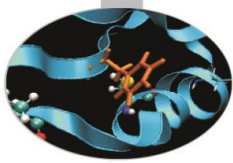
```
! Stampa della matrice [BT]

 write(unit=10, fmt="(a)") " Stampa della matrice
[BT] "
 write(unit=10, fmt="(a)") " "

 do i = 1, l2
    write(unit=10, fmt="(10 (f10.5,2x))") BT(i,:)
 end do ! i

 write(unit=10, fmt="(a)") " "
```

Soluzioni



Esercizio 3/5

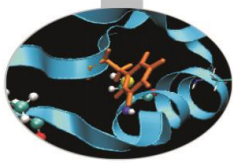
```
! Stampa della matrice [C]

 write(unit=10, fmt="(a)") " Stampa della matrice
[C] "
 write(unit=10, fmt="(a)") " "

 do i = 1, l1
    write(unit=10, fmt="(10 (f10.5,2x))") C(i,:)
 end do ! i

 write(unit=10, fmt="(a)") " "
```

Soluzioni



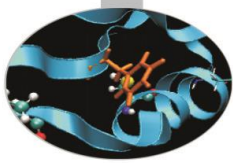
Esercizio 3/6

```
 ! Stampa della matrice [A]

 write(unit=10, fmt="(a)") " Stampa della matrice
[A] "
 write(unit=10, fmt="(a)") " "

 do i = 1, l2
    write(unit=10, fmt="(5 (f10.5,2x))") A(i,:)
 end do ! i
```

Soluzioni



Esercizio 3/7

```
! Valutazione del tempo finale
 call date_and_time(values=Tf)
 secf = Tf(5)*3600.0 + Tf(6)*60.0 + Tf(7)*1.0 + &
        Tf(8)*0.001
! Stampa a video dei tempi (totale, di calcolo e di
!scrittura)
 write(unit=*,fmt=*) "tempo totale impiegato: ",(secf&
                     -sec0)
 write(unit=*,fmt=*) "tempo di calcolo : ",(secc-sec0)
 write(unit=*,fmt=*) "tempo di write (formattato con &
                    &  un ciclo do): ",(secf-secc)
 end program Test_1
```