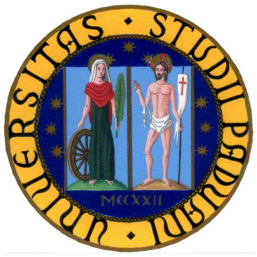
I/O in Fortran

Paolo Bison

Fondamenti di Informatica
A.A. 2006/07
Università di Padova



Print e Read

- sintassi

```
R911    print-stmt
    is   PRINT format [ , output-item-list ]
R909   read-stmt
   is  READ ( io-control-spec-list )
                  [ input-item-list ]
   or  READ format [ , input-item-list ]
R913    format
    is   char-expr
    or   *
```

Formato di I/O

- specificazione del formato di scrittura/lettura
- stringa costante/variabile di tipo character "`format-specification`"

```
R1002     format-specification
      is      ( [ format-item-list ] )
R1003     format-item
      is      [ r ] data-edit-desc
      or      control-edit-desc
      or      [ r ] ( format-item-list )
R1005     data-edit-desc
      is      I w [ .  m ]
      or      F w .  d
      or      ES w .  d
      or      E w .  d
      or      L w
      or      A [ w ]
R1010     control-edit-desc
      is       n X
      is       T c
      or      [ r ] /
```


Parametri dei descrittori

- `r`
  fattore di ripetizione
- `w`
  ampiezza di campo
- `m`
  numero minimo cifre da visualizzare
- `d`
  numero cifre decimali

Descrittori dati

- `Iw [.m]`
  valore intero
- `Fw.d`
  valore reale in notazione decimale
- `ESw.d` e `Ew.d`
  valore reale in notazione scientifica $m \times 10^e$
  ES: $1.0 \leq m < 10.0$
  E: $0.1 \leq m < 1.0$
- `Lw`
  valore logico
- `Aw`
  stringhe di caratteri

Descrittori di controllo

- `nX`
  salta n spazi
- `Tc`
  avanza alla posizione c della riga
- `/`
  va a nuova linea
- `'chars'`
  stampa i caratteri chars
- esempio d'uso dei descrittori
  io_test.f90

I/O su file

- istruzioni su file

```
R216      action-stmt
      is .....
      or      open-stmt
      or      read-stmt
      or      write-stmt
      or      close-stmt
      or  .....
```


Open

- associazione tra un file ed un numero di unità
- sintassi

```
R904      open-stmt
      is      OPEN ( connect-spec-list )

R905      connect-spec
      is      UNIT = file-unit
      or      IOSTAT = int-variable
      or      FILE = file-name-expr
      or      STATUS = scalar-char-expr
      or      ACTION = scalar-char-expr
```

Parametri dell’open

- UNIT
  valore intero da associare al file
- IOSTAT
  variabile intera per il codice d’errore
  0 operazione OK, $\neq$ 0 operazione errata
- STATUS
  una tra "old", "new", "scratch", "replace"
- ACTION
  una tra "read", "write", "readwrite"

Close

- chiude il file associato ad un valore di unità
- sintassi

```
R907      close-stmt
      is      CLOSE ( close-spec-list )

R908      close-spec
      is      UNIT = external-file-unit
      or      IOSTAT = int-variable
      or      STATUS = char-expr
```

- STATUS
  una tra "keep", "delete"

Read e write

- sintassi

```
R909      read-stmt
      is      READ ( io-control-spec-list ) [ input-item-list ]
R910      write-stmt
      is      WRITE ( io-control-spec-list ) [ output-item-list ]
R912      io-control-spec
      is      UNIT = io-unit
      or      FMT = format
      or      IOSTAT = scalar-default-int-variable
```

- * valore default
- equivalenze

  print *,items ≡ write (*,*) items ≡ write (unit=*,fmt=*) items

  read *,items ≡ read (*,*) items ≡ read (unit=*,fmt=*) items

write_file.f90

```
program write_file
integer :: i,err,n_min,n_max,n
integer :: rand_int
real :: rnd
character (len=50) :: f_name
print *,"file name"
read *,f_name
print *,"n_min n_max"
read *,n_min,n_max
print *,"n"
read *,n
```

write_file.f90

```
! inizializza generatore numeri casuali
call random_seed() !
open(unit=8,file=trim(f_name), &
      iostat=err,status="replace",action="write")
if (err/=0) then
  print *,"impossibile creare un file"; stop
end if
do i = 1,n
  ! ritorna un valore reale tra 0 e 1
  call random_number(rnd)
  rand_int = int((n_max - n_min + 1)*rnd) + n_min
  write (unit=8,fmt=*,iostat=err)rand_int
end do
close(unit=8)
end program write_file
```


read_file.f90

```
program read_file
integer :: err,num
integer :: somma
character (len=50) :: f_name
print *,"file name"
read *,f_name
open(unit=8,file=trim(f_name), &
      iostat=err,status="old",action="read")
if (err/=0) then
  print *,"file non esiste"
  stop
end if
```

read_file.f90

```
somma=0
do
  read(unit=8,fmt=*,iostat=err)num
  if (err/=0) then ! fine file
    exit
  end if
  somma = somma+num
end do
close(unit=8)
print *,somma
end program read_file
```


Do implicito

- ciclo iterativo come argomento di istruzioni I/O

```
R434     ac-implied-do
     is      ( ac-value-list , ac-implied-do-control )
R435     ac-implied-do-control
     is      ac-do-variable = int-expr , int-expr [ ,int-expr ]
R436     ac-do-variable
     is      int-variable
```

- esempi

```
print "(10('a=',I3,/))",(a(i),i=10,1,-1)
print "(10(I2,' a=',I3,' b=',I3,/))", &
        (i,a(i),b(i),i=10,1,-1)
```

Normalizzazione di valori

- dato un insieme di valori memorizzati in file creare un file contenente i corrispondenti valori normalizzati tra -1 e 1
- max(abs(x)) = 1
- file norm_file.f90