Fortran

Chapter 8 檔案

檔案讀取可分為”循序讀取”及”直接讀取”兩種情形：

(1) 循序讀取：對一個檔案在讀入或者是寫出時，我們只能從頭開始，一步步地向下來一筆一筆地讀取資料。

(2) 直接讀取：對一個檔案在讀入或者是寫出資料時，我們可以任意、直接地跳躍到檔案的任何一個位置上來從事讀取的工作。

檔案儲存模式分為”文字檔”及”二進位檔”

(1) 文字檔：把所有的資料都用我們人眼可以明白理解的字元符號來做儲存。優點：易懂，可直接修改。

(2) 二進位檔：直接把資料在電腦記憶中的儲存情形(也就是二進位碼)直接寫入檔案中。優點：讀取較快速、省空間。

8-1-1 The open statement

OPEN(unit = int_expr, file = char_expr, status = char_expr, action = char_expr, iostat = int_var)

(1) unit = int_expr：開啟一個檔案時要給定這個檔案一個讀取的編號，以後使用 write, read 時使用這個編號就可以對這個檔案來讀寫了。

unit = int_expr 的值最好避開 1, 2, 5, 6。2, 6 是指內定的輸出位置，也就是螢幕。1, 5 則是捏內定的輸入位置，也就是鍵盤。

(2) file = char_expr：用來指定開啟檔案的名稱。

(3) status = char_expr：'New', 'OLD', "Scratch' or 'unknown'用來標示是要開啟一個新檔或是已經存在的舊檔

states = 'New'：這個檔案原本不存在，是第一次開啟

states = 'OLD'：這個檔案原本就已經存在

(4) Action = char_expr : 'read', 'write', 'readwrite'

Action = 'readwrite'： 表示所開啟的檔案可以用來讀取及寫入，這是內定值

Action = 'read'： 表示所開啟的檔案只能用來讀取資料

Action = 'write'： 表示所開啟的檔案只能用來寫入資料

(5) Iostate = int_var： 表示檔案開啟的狀態

int_var > 0 表示讀取動作發生錯誤

int_var = 0 表示讀取動作正常

int_var < 0 檔案終了

(6) Access = 'sequential' or 'direct'

Access = 'sequential' 讀取檔案的動作會以"循序"的方法來做讀取

Access = 'direct' 讀取檔案的動作可以壬意指定位置

(7) Position = 'asis' or 'rewind' or 'append'

Position = 'asis' 表示檔案開啟時的讀取位置，不特別指定。(內定值)

Position = 'rewind' 檔案開啟時的讀取位置移到檔案的開頭處。

Position = 'append' 檔案開啟時的讀取位置移到檔案的結尾處。

Case 1: opening a file for input

```
integer :: ierror
 open (unit = 8, file = ‘EXAMPLE.DAT’, status = ‘OLD’, action = ‘read’, iostat = ierror)
```

Case 2: opening a file for output

```
integer ::n_unit, ierror
character(len = 6) :: filename
n_unit = 25
filename = ‘outdat’
 open (unit = n_unit, file = filename, status = ‘new’, action = ‘write’, iostat= ierror)
```

Case 3: opening a scratch file

```
 open (unit = 12, status = ‘scratch’, iostat= ierror)
```

8-1-2 The close statement

close (close_list)

8-1-3 reads and writes to disk files

(1) open(unit = 8, file = ‘input.dat’, status = ‘old’, iostat = ierror)

read(8, *) x, y, z

~read the values of variables x, y and z from the file “input.dat”.

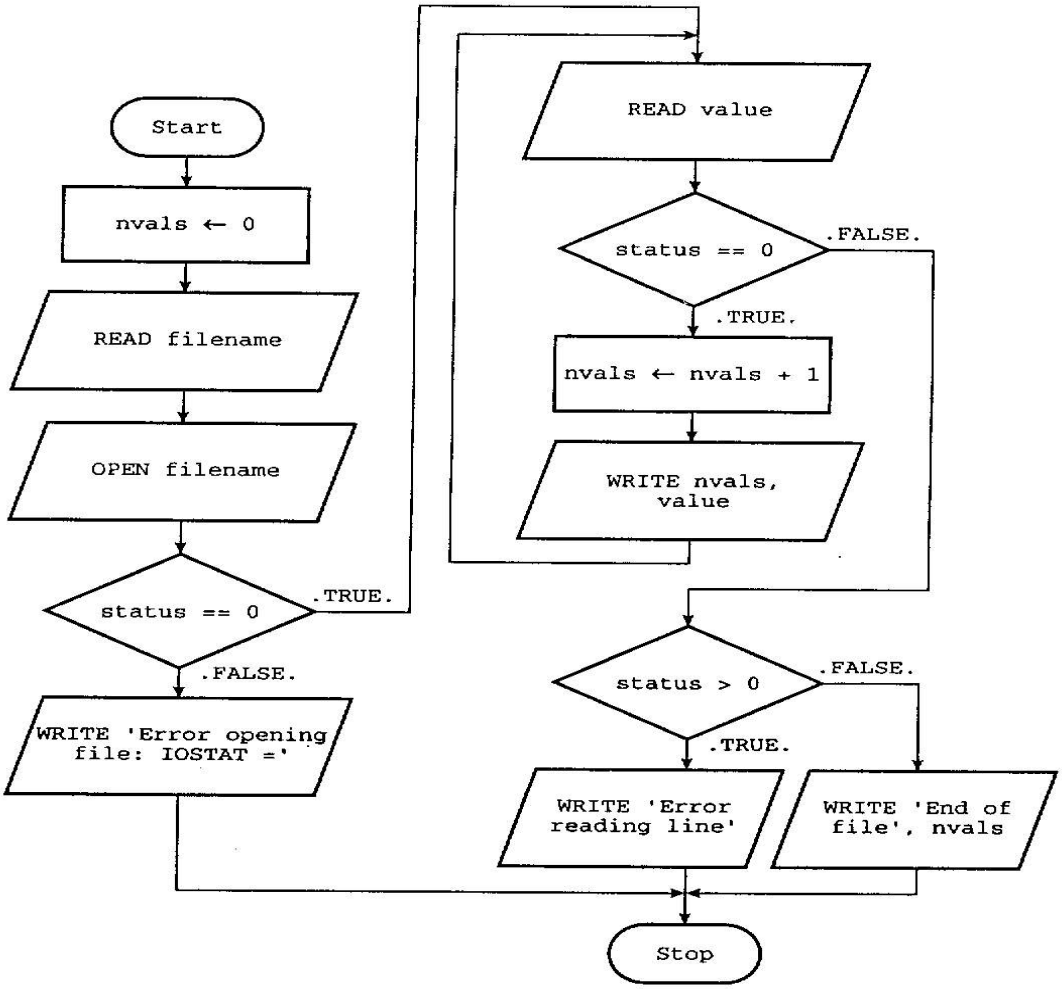
(2) open(unit = 9, file = ‘output.dat’, status = ‘new’, iostat = ierror)

read(9, 100) x, y, z

100 Format(‘X = ‘, F10.2, ‘Y=’, F10.2, ‘Z=’, F10.2)

~write the values of variables x, y and z to the file output.dat.

Example: Reading data from a file:



Flowchart for a program to read an unknown number of values from an input data file.

```
Program read
      implicit none
      character (len =20) :: filename
      integer :: nvals = 0
      integer :: status
      real :: value

      ! Get the file name and echo it back to the user.
      write(*,*) 'Please enter input file name:'
      read(*,*) filename
      write(*,10) filename
10    format(1x, 'The input file name is : ', A)

      ! Open the file and check for errors on open
      open(3, file = filename, status = 'old', action = 'read', iostat = status)
      openif: If(status == 0) then
         ! open was OK. Read values
         readloop: do
            read(3, *, iostat = status) value
            if(status /= 0) Exit
            nvals = nvals + 1
            write(*, 20) nvals, value
20          format(1x, 'Line', I6, ': Value = ', F10.4)
         end do readloop

         ! The while loop has terminated. Was it because of a read error or because
         ! of the end of the input file?
         readif: if(status > 0) then
            write(*, 30) nvals + 1
30          format(1x, 'An error occurred reading line', I6)
         else
            write(*, 40) nvals
40          format(1x, 'End of file reached. There were ', I6, 'values in the file.')
         end if readif
         else openif
         write(*, 50) status
50       format(1x, 'Error opening file: IOSTAT = ',I6)
         end if openif
         close(3)
 end program read
```

read(3, *, iostat = status) value

status > 0 表示讀取動作發生錯誤

status = 0 表示讀取動作正常

status < 0 表示檔案終了

直接存取檔的操作

把檔案的空間、內容事先加以分割成一個個同樣大小的小區塊，並且把這些區塊按順序加以編號。而讀寫檔案時，要先指定檔案讀寫位置要在那一個區塊上，才能進行讀寫的工作。

直接存取檔可以任意在檔案的任何一個地方來進行讀寫的工作。

Example:

“兄弟象”在一場棒球比賽中的打擊者打擊率依棒次順序列表在檔案 List 中如下：

3.12

2.98

3.34

2.86

2.54

2.78

2.23

2.56

請寫一個可以由棒次來查尋打者打擊率的程式。

```
Program ex0909
  implicit none
  character(len = 20), parameter :: input = 'List'
  integer, parameter :: players = 9
  integer :: player
  integer, parameter :: rec_length = 6
  real :: hit_rate
  open(10, file = input, form = 'formatted', access = 'direct', & recl = rec_length)
  do while (.true.)
    write(*,*) 'Number:'
    read(*,*) player
    if(player < 1 .or. player > players) exit
    read(10, fmt = '(F4.2)', rec = player) hit_rate
     write(*, 100) 'Number ', player, 'hit_rate = ', hit_rate
100   format(1X, A8, I2, A10, F5.2)
  end do
  stop
  end program ex0909
```

讀取檔案位置

(1) 開啟直接讀取檔時，open 敘述中的 access = 'direct'及 recl 後的數值不能省略。這個數值是用來切分出檔案區塊大小使用的。

(2) 在 DOS 作業系統中，文件檔中每一行的行尾都有兩個看不見的符號用來代表一行文字的結束。所以真正一行的長度就是"一行文字字元的數量再加上 2"

e.q. 在 List 檔中每行長度 = 4 + 2 = 6

在 unix 中，每一行的行尾只需一個結束符號，所以一行的長度就是"一行文字字元的數量再加 1"

Example : 依選手的背號順序，輸入選手的打擊率

```
Program ex0910
  implicit none
  character(len = 20), parameter :: input = 'newList'
  integer, parameter :: players = 9, rec_length = 6
  integer :: player
  real :: hit_rate
  open(10, file = input, form = 'formatted', access = 'direct', & recl = rec_length)
  do while (.true.)
    write(*,*) 'Hit Number:'
    read(*,*) player
    if(player < 1 .or. player > players) exit
    read(10, fmt = '(F4.2)', rec = player) hit_rate
    write(*,*) 'Input hit rate:'
    read(*,*) hit_rate
    write(10, fmt = '(F4.2)', rec = player) hit_rate
  end do
  stop
  end program ex0910
```

執行結果:

```
Hit Number : 3
Input hit rate : 2.54
Hit Number : 5
Input hit rate : 3.46
Hit Number : 2
Input hit rate : 3.44
Hit Number : 0
```

Newlist 檔案

□ □ □ □ □ □3 · 4 4□ □2 · 5 4□ □ □ □ □ □ □ □3 · 4 6□ □

二進位檔的操作

二進位檔：肉眼無法明白了解之亂碼檔

Example: 把輸入棒球選手打擊率的程式，改成使用二進位檔來運作：

```
Program ex0911
Implicit none
character(len = 20), parameter :: output = 'List.bin'
integer, parameter :: players = 9, rec_length = 4
integer :: player
real :: hit_rate
 open(10, file = output, form = 'unformatted', access = 'direct', & recl = rec_length)
do while (.true.)
   write(*,*) 'Hit Number :'
   read(*,*) player
   if (player < 1 .or. player > players) exit
   read(*,*) hit_rate
   write(10, rec = player) hit_rate
end do
stop
end program ex0911
```

hit_rate 是單精度

可節省儲存的空間，儲存單精度的浮點數只需 4 個 bytes。若以文字檔來儲存同樣精準度的浮點數，其所需之 byte 遠超過 4 個 bytes。因此，如果要存放"精確"及"大量"的資料時，使用二進位檔案是比較好的選擇。