FORTRAN

除錯（debug）、常見疑難雜症排解、程式設計技巧

範例檔：
/home/teachers/weitingc/lecture_ex/test_dsq.f95, dsq_int.txt

良好的習慣有助「防蟲」

- 動手寫程式前，先花時間寫好草稿、列出會使用的變數/常數，要寫為函式或副程式的部分等
  - 構思的時間比動手寫的時間多，是很正常的
- 不要等到整個程式都寫完才編譯、測試。寫完一段，就可以先進行測試，確定無誤再往下寫。
  - 不要讓蟲有大量繁殖的機會…
- 邊寫就邊加上註解，可以幫助思考與記錄想法。
- 先寫下對稱平衡的語法（PROGRAM/END PROGRAM, IF/ENDIF, DO/ENDDO, ( )），再填入中間的程式
  - 可在ENDIF或ENDDO後加上註解，方便找到對應的開頭
- 維持整潔的排版：縮排、空行、大小寫使用的習慣
  - 「紀律」與「潔癖」是防蟲的第一步。

良好的習慣有助「防蟲」(2)

- 一定要加「IMPLICIT NONE」（主、副程式都要）。
- 「常數」務必宣告為常數。
- 制訂變數取名的規則、並且賦予恰當的初始值。
- 盡可能以常數控制陣列大小與迴圈次數。
- 程式中若出現除法，務必思考程式執行過程中是否會發生分母為零的狀況。如果會，記得增加判斷式，讓分母為零時仍可順利運作。

```
IF (b/=0) THEN
   dsq=a/(b**2)
ELSE
   dsq =-999.9
   WRITE(*,*) 'b=0'
ENDIF
```

- 同一支程式中，固定counter的使用習慣
  - （例如i固定用來處理緯度相關的維度、k處理高度相關維度等）

良好的習慣有助「防蟲」(3)

- 將編譯的指令寫成shell script，避免程式檔被執行檔覆蓋的「慘劇」
  - 開啟一個名為compile的文字檔
  - 寫入編譯的指令（確保指令是完全正確的）

```
F95 -o abc.exe aaa.f95 bbb.f95 ccc.f95
```

  - 存檔後，將compile的權限改為可執行

```
chmod +x compile
```

  - 執行shell script，就可以完成編譯程式的步驟

```
./compile
```

儘管再小心，程式還是會「長蟲」

- 就算是最有經驗的programmer，也很少能一次就寫出完全「無蟲」的程式。「除蟲」（debug）往往比寫程式本身花的時間多。
- 抓蟲必備：耐心、清醒的頭腦
- 「蟲」的種類很多，很難全部列出，下面是一些常見的品種與可能的症狀：

- **Compiler可以抓出的蟲（善良的蟲？）**

症狀：編譯不過，編譯器會列出錯誤

病因：通常跟「語法」有關（syntax error）

例如：拼錯字、少括號、少ENDIF、
變數未宣告（若有加IMPLICIT NONE）、
未正確呼叫副程式或函式
…

程式「蟲」的種類(2)

- **Compiler抓不出的蟲（壞蟲？）**

編譯可以過關！但程式執行時會有問題，又分三種：

（第一種）

症狀：程式無法順利執行完畢

病因：若出現runtime error訊息，通常跟「檔案」、讀取或輸出的變數種類與個數有關

例如：檔案不存在、
程式中檔名錯誤、
檔案資料長度不對、
讀取的資料種類不對、
輸出格式與變數種類不符

病因：若出現segmentation fault訊息，則是與記憶體有關

例如：使用陣列時，超過原本陣列宣告的長度

程式「蟲」的種類(3)

- **Compiler抓不出的蟲（壞蟲？）**

編譯可以過關！但程式執行時會有問題，又分三種：

（第二種）

症狀：程式可順利執行，但數值跟預期相比天差地遠（或出現Inf）

病因：通常是誤用整數變數來處理實數運算、
主程式副程式傳遞的變數種類不一致、
誤把陣列當單一變數、
使用錯誤的陣列元素、
整數數值過大溢位（overflow）、
除式分母為零、
四則運算錯誤（例如：*與**搞錯、括號放錯、弄錯變數）
…

「溢位」問題

- 超過數值能夠被記憶體正確儲存的數值範圍，就會發生「溢位 overflow」問題（出現 Inf！）
- 一般機器用32bit來儲存一個數值，能夠正確儲存的範圍：
  - 實數約在 $(-3.4\times10^{38})\sim(+3.4\times10^{38})$ 之間
  - 整數在 $(-2^{31})\sim(+2^{31}-1)$ 之間
- 若希望程式能處理大範圍的實數數值，可以將實數變數或陣列宣告為「倍精度」，可正確儲存$(-1.7\times10^{308})\sim(+1.7\times10^{308})$ 之間的實數
- 在支援64bit的機器上，可以將整數變數宣告為integer(kind=8)，用64bit儲存一個整數，就可正確儲存$(-2^{63})\sim(+2^{63}-1)$ 之間的整數，然而這個程式就不一定在所有機器都適用。

「溢位」問題

- integer變數處理絕對值很大的正整數或負整數（例如：10**10, -EXP(25.2) ）會出現錯誤（數值錯誤，甚至正數還變負數！？）

- 用4bit來舉例說明：一個bit（位元）只會有0或1兩種狀態，若用4bit來儲存一個整數，則最左邊的bit用來儲存正負號，0=正，1=負，其餘bits代表數值
  - 數值0: 0000　　數值2: 0010　　數值3: 0011…　數值7: 0111
  - 二進位的負數表示方法，是正數的補數加一
    數值-3: 1101　數值-7: 1001　數值-8: 1000
  - 4bit能夠正確儲存的數值範圍是 (-2^3)~ $(+2^3-1)$ = -8~7

- 若因溢位改變最左邊位元的值（0 <-> 1），顯示的數值與正負號就會有問題
  - 想要儲存數值8： 由7 (0111) +1進位
    → 1000 **overflow!! 顯示數值 -8**

程式「蟲」的種類(4)

- **Compiler抓不出的蟲（壞蟲？）**

編譯可以過關！但程式執行時會有問題，又分三種：

（第三種）

症狀：程式可順利執行，數值看似合理但不正確

病因：通常是數值（numerical）問題 或 記憶體問題

例如：所進行的運算結果無法以單精度處理、
未在程式內賦予初始值（導致執行時系統填入預期外的初始值）、
判斷實數值「相等」時未考慮數值誤差
…

單精度 vs. 倍精度

- 記憶體以二進位的方式儲存實數（又稱浮點數float），當數值沒辦法用二進位精準表達時，會出現微小的「進位誤差」（rounding error）
- Fortran宣告的實數預設為「單精度」(single precision)，使用32bits儲存一個實數，最多能夠精確至小數點後8位，之後位數會被捨去，造成微小的捨去誤差（round-off error）
- 需要更高精確度時，可以改用double precision來宣告「倍精度」實數，此時會使用64bits來儲存，可精確度至小數點後16位，缺點是較佔記憶體。

單精度 vs. 倍精度

- 宣告為倍精度的實數變數，賦值時要記得在末尾加上D0，讓後面所有的位數設為零，以避免rounding error。

```
PROGRAM demo
real :: x
double precision :: y, z

x = 1.1
y = 1.1
z = 1.1D0
WRITE(*, *) "x =", x, " , y =", y, " , z =", z

END PROGRAM
```

輸出結果：

x = 1.1000000**2** ， y = 1.1000000**2** ， z = 1.1

x：宣告為單精度，二進位無法精確表達數值1.1，因此小數點下第8位出現誤差
y：宣告為倍精度，但未用倍精度格式賦值，因此在小數點下第8位仍出現誤差
z：宣告為倍精度，並且使用倍精度格式賦值，因此在小數點下第2位以下均為0

判斷實數數值「相等」的注意事項

- 電腦能夠儲存的實數精確度有限，在進行運算時，會產生一些微小的誤差。進行實數數值相等的邏輯判斷時（如x==3.0），必須考量這個狀況。建議設定一個可接受的誤差範圍，只要小於此範圍，代表實數數值相等。舉例：

```
...
IF ( x == 3.0 ) THEN
...
```

因為精確度限制， 運算後x的數值可能變成2.999999或3.000001，無法滿足 x==3.的條件

vs.

```
...
IF ( ABS(x - 3.) <= 1.E-6 ) THEN
...
```

設定誤差範圍為10^{-6}，當x的數值在2.99999~3.000001之間，就符合條件。

萬用抓蟲法：WRITE(*,*)

- 如果程式可以編譯，但無法順利執行，或者執行結果有錯，可以在程式中加入臨時的輸出指令，把運算中變數的數值顯示在螢幕上，找出不合理的數值在程式的何處發生。
- 可加上註解，標記臨時的輸出指令為debug專用
- 除完蟲之後，可以用**!**「關掉」這些輸出指令（不要直接刪除，方便日後需要時可快速「打開」）

```
...
DO I = 1, 10
  X(I) =
    A(I)**2
ENDDO
...
```

```
...
! Debug output start
  WRITE(*,*) 'before loop:', X, A
! Debug output end
DO I = 1, 10
  X(I) = A(I)**2
! Debug output start
  WRITE(*,*) 'inside loop:', I, X(I), A(I)
! Debug output end
ENDDO
...
```

萬用抓蟲法：WRITE(*,*) (2)

- 也可以利用邏輯變數與判斷式做為debug輸出的「開關」

```
Logical :: debug=.T.
...
! Debug output start
IF (debug) THEN
  WRITE(*,*) 'before do loop:', X, A
ENDIF
! Debug output end
DO I = 1, 10
  X(I) = A(I)**2
! Debug output start
IF (debug) THEN
  WRITE(*,*) 'inside do loop:', I, X(I), A(I)
END
! Debug output end
ENDDO
...
```

日後只要改為debug = .F.，再重新編譯程式，執行時就不會有臨時的輸出

結語

- 這邊只列出常見的錯誤或疑問，實際遇到的狀況可能更多元，尤其是當多個錯誤同時存在於程式中的時候，可能出現的「症狀」會更複雜。
- 以初學者來說，最重要的原則就是寫完一小段程式就先測試，移除錯誤後再繼續往下寫，避免bug的累積。
- 正面思考：debug的經驗也是學習程式很重要的一部份，尋找、移除bug的各種嘗試，會讓你更熟悉fortran與電腦的運作。

常見錯誤

- 宣告
  - 種類宣告錯誤
  - 忘記宣告且未加IMPLICIT NONE
- 賦值
  - 變數種類與所賦予的值種類不一致
- 運算、語法錯誤
  - 拼錯字、標點、運算符號、 不平衡的語法（’’, (), IF…ENDIF）
- 陣列subscript
  - 超出陣列所宣告的大小，或維度錯誤
- I/O
  - 未開啟檔案、開啟/讀寫的檔案不存在
  - 讀取超過檔案長度
  - 讀寫的數值（或格式碼）與變數種類不一致
- Function/subroutine
  - 傳遞的變數種類、數目不一致
  - 未加RETRUN/END
- 編譯
  - 副程式、函式未與主程式一起編譯
  - 覆蓋掉程式碼

Self-Learning Resources: FORTRAN 90/95

- 網頁：
  - **Fortran 90/95 reference** http://www.icl.utk.edu/~mgates3/docs/fortran.html
  - **Fortran 內建函數** https://gcc.gnu.org/onlinedocs/gfortran/Intrinsic-Procedures.html#Intrinsic-Procedures
  - **Fortran Online Resources** http://www.lahey.com/other.htm

- 參考書:
  - **Chivers, I, and J Sleightholme (2012), Introduction to Programming with Fortran, Springer [台大圖書館電子書] http://link.springer.com/book/10.1007/978-0-85729-233-9/page/1**
  - Chapman, S. (2004), Fortran 90/95 for scientists and engineers, 2nd Ed., McGraw-Hill Higher Education
  - Press, W. H., et al. (1997) Numerical Recipes in Fortran 90: The Art of Parallel Scientific Computing, Cambridge
  - 彭國倫 (2001), Fortran 95程式設計 (ISBN:9575669592)

下週 11/02 期中考

- 範圍：第一週到本週與FORTRAN相關的課程、影片、範例檔與作業
- 個人上機考試（3:30PM~5:30PM）
- OPEN BOOK (NOTES, INTERNET, HOMEWORK…)
- 考試期間不可與旁人討論或傳遞訊息、檔案（違規者取消考試資格）
- 使用自己的筆電
  （需要借用筆電請在本週六前先預約）
- 佔原始總成績10%

期中考準備

- 重點觀念：
  - 變數種類與宣告方式（實數vs整數，常數，陣列）
  - 邏輯判斷的語法與使用時機
  - 迴圈（語法、counter在迴圈執行過程中如何變化）
  - 陣列（如何宣告不同大小的陣列、陣列的元素、subscript的語法、迴圈與陣列的搭配、陣列的內建函數SUM, COUNT, MAXVAL…）-- review.f95
  - 讀取檔案、輸出檔案的語法
  - Function（與主程式之間如何傳遞變數）
  - Subroutine（主程式如何呼叫subroutine，與主程式之間的變數傳遞）
- 瞭解範例檔與作業程式