Fortran Week 2

- HW10解答：
  /home/teachers/fortran_ta/data/PSC2020/hw10/

- 本週影片 問題 (Do Loop, If, 1D array, file I/O)
- 二進制與溢位
- hw11作業說明

容易踩到的地雷(1)

- 錯誤：宣告區與執行區交錯→編譯時出現error message，舉例

```
IMPLICIT NONE
CHARACTER (17) :: out1
out1='The lapse rate is'
CHARACTER (6)  :: out2
…
```

一行的開頭直接寫 變數名 = 值
是「執行區」的語法！**XXXX**
Compile error message:
Error: Unexpected data declaration statement

- 修正：

```
IMPLICIT NONE
CHARACTER (17) :: out1='The lapse rate is'
CHARACTER (6)  :: out2
```

宣告時順便賦值，**ok!**
這是屬於宣告區的語法

或

```
IMPLICIT NONE
CHARACTER (17) :: out1
CHARACTER (6)  :: out2
out1='The lapse rate is'
```

「宣告區」全部結束後
才進入「執行區」，**ok!**

皆可

容易踩到的地雷(2)

- 錯誤：「讀取變數數值」與「變數運算」的順序錯誤
  - Fortran的執行是一行一行依序往下（除非有迴圈才會反覆，或是有if判斷，才會跳過某幾行）
  - 運算式的計算，會使用變數在之前被賦予的值代入。
  - 如果變數沒有被賦值，一般會用0代入（但可能隨著compiler而有所不同）
- 舉例

```
Real:: z1, z2, Z
Z=(z1+z2)/2.
WRITE(*,*) 'Please input z1 and z2 in meter:'
READ(*,*) z1, z2
WRITE(*,*) Z
```

z1,z2之前都沒被賦值，用0代入算出Z (=0)

T到這邊才被賦值

0

- 修正：

```
Real:: z1, z2, Z
WRITE(*,*) 'Please input z1 and z2 in meter:'
READ(*,*) z1, z2
Z=(z1+z2)/2.
WRITE(*,*) Z
```

z1, z2被賦值（鍵盤輸入）

用使用者輸入的z1, z2值代入，算出Z

正確的Z值

課堂示範：**(1) if**

- /home/teacher/weitingc/work/class2a.f95

- 接收使用者輸入實數變數T（絕對溫度）
- 如果T≥0，則在螢幕顯示T-273.15的結果；
  其餘情況，則在螢幕顯示 'T out of range'

- 編譯為class2a.exe

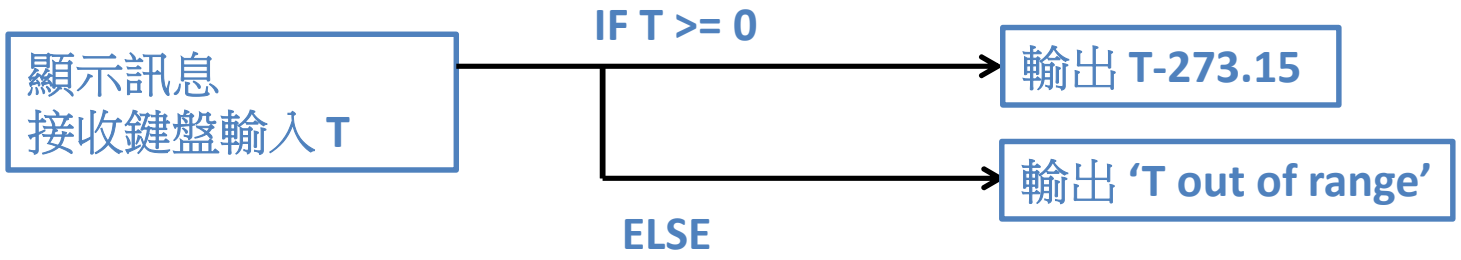
課堂示範：(2) do loop

- /home/teacher/weitingc/work/class2b.f95

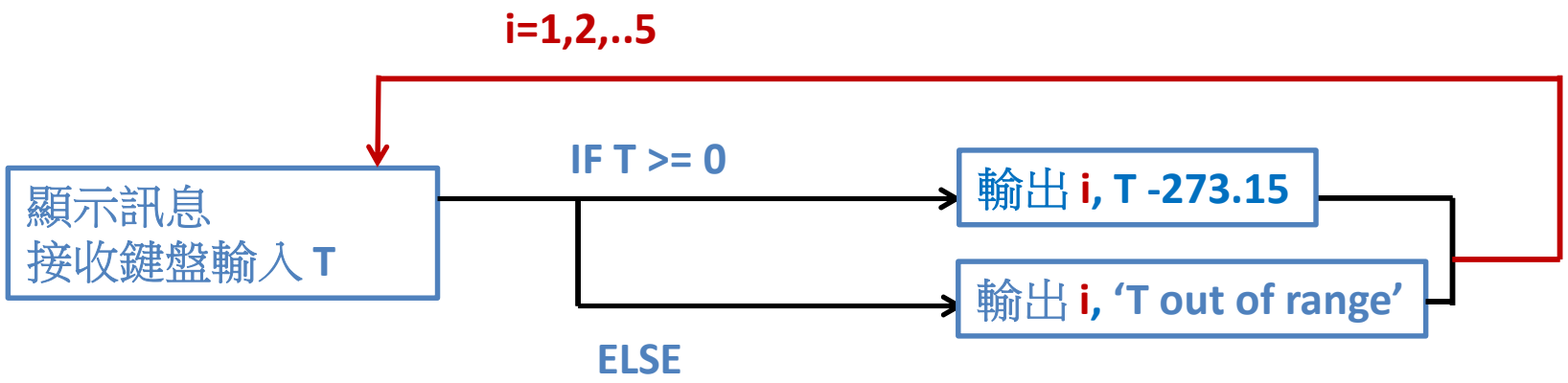
- 用迴圈讓下面這段重複五次：
  - 接收使用者輸入實數變數T（絕對溫度），
  - 如果該次輸入的T≥0，則在螢幕顯示「次數（整數）」與該次輸入的T-273.15結果；
    其餘情況，則在螢幕顯示「次數」與’T out of range’

- 編譯為class2b.exe

class2a.f95



class2b.f95



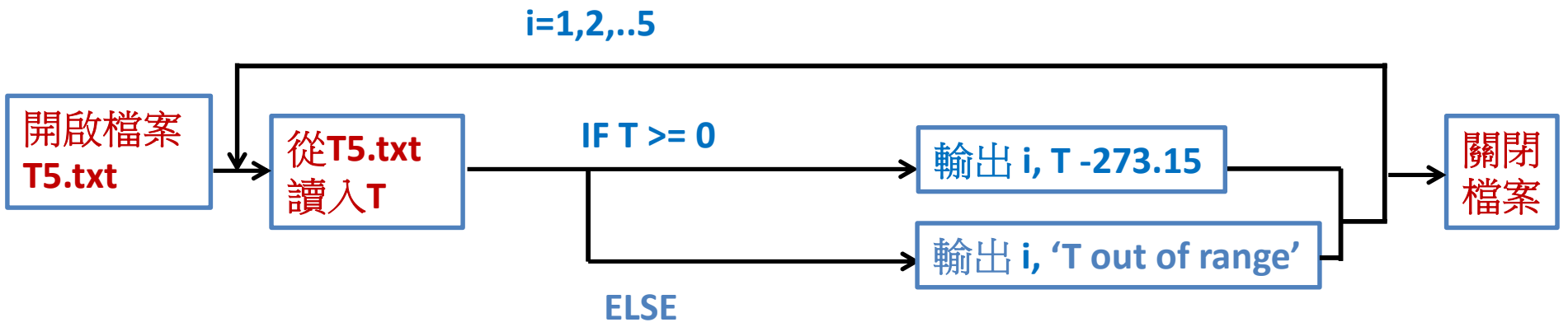
課堂示範：(3) read from file

- /home/teacher/weitingc/work/class2c.f95 & T5.txt

- 用迴圈讓下面這段重複五次：
  - 讀取T5.txt當中的一個數字，為實數變數T賦值，
  - 如果該次讀取的T≥0，則在螢幕顯示「次數（整數）」與該次讀取的T-273.15結果；
    其餘情況，則在螢幕顯示「次數」與’T out of range’

- 編譯為class2c.exe

class2c.f95

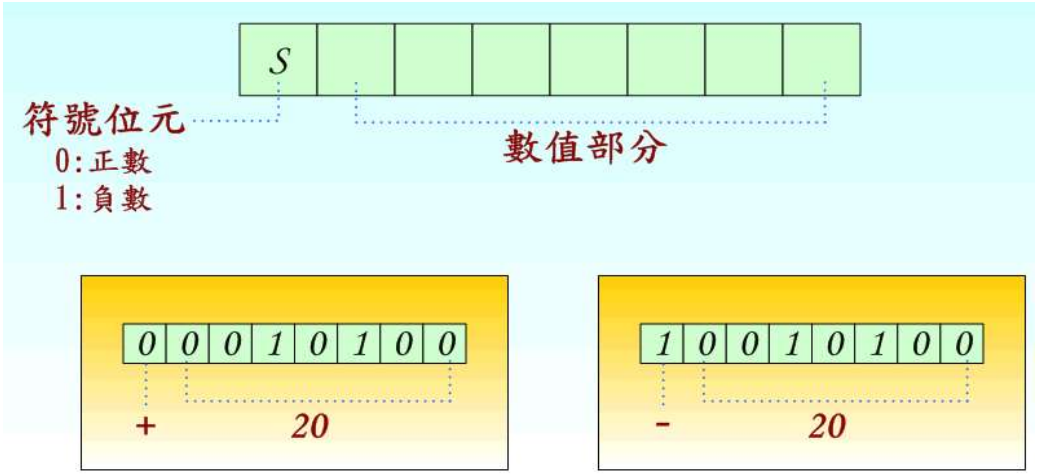

i=1,2,..5
開啟檔案
T5.txt
從T5.txt
讀入T
IF T >= 0
輸出 i, T -273.15
輸出 i, 'T out of range'
ELSE
關閉
檔案

電腦儲存數值的方法(二進制)與溢位(overflow)

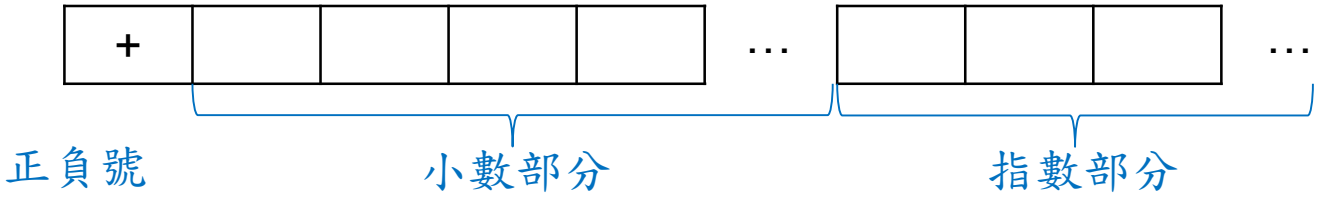
- 以下講義摘錄修改自 http://research.ydu.edu.tw/~jimmy/9501imbcc/ch02.ppt

能夠儲存數值的位元數有限→能表達的數值範圍有限

- 用其中一個位元(通常在最左邊) 來表示該數值為正數還是負數
  - 使用n個位元時，數值的表達範圍只剩n-1個位元可以使用
  - 正整數的表達範圍是+0~+(2^{n-1}-1)
  - 負整數的表達範圍是-(2^{n-1}-1)~-0
  - 使用帶符號大小表示0的時候，+0與-0是不一樣的。
- 以8位元來表示正負整數：範圍-128～+127。



實數的儲存方式 (32位元的機器為例)



- 小數部分占的位元數愈多，表達的有效數字愈多，精度愈高
- 指數部分占的位元數愈多，則能表示的數值範圍愈大。

- 單精度 (32位元儲存一個實數) float
  - 有效數字6~7，範圍3.4E-38～3.4E+38
- 倍精度 (64位元儲存一個實數) double
  - 有效數字15~16，範圍1.7E-308～1.7E+308

二進制的加法運算

被加數	+	加數	=	結果
0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	0，但進位1

- 舉例: 二進制1010110與100011的加法運算，並使用8個位元來存放：
  - Step1：被加數與加數不足位數的部分先補0。
  - Step2：由右至左，開始進行加法。(右邊倒數第二位遇到進位)
  - Step3：不斷由右至左進行加法與進位，最後結果為1111001

```
  01010110
+ 00100011
----------
```

```
進位      1
  01010110
+ 00100011
----------
        01
```

```
進位     11
  01010110
+ 00100011
----------
       001
```

```
進位     11
  01010110
+ 00100011
----------
  01111001
```

溢位 (overflow)

- 當相加或相減的結果超過位元上限時，會發生「溢位」現象。舉例:
  - 8個位元可表達的整數範圍是 -128～+127。
  - 計算十進位的+127+1（二進位的01111111+1）

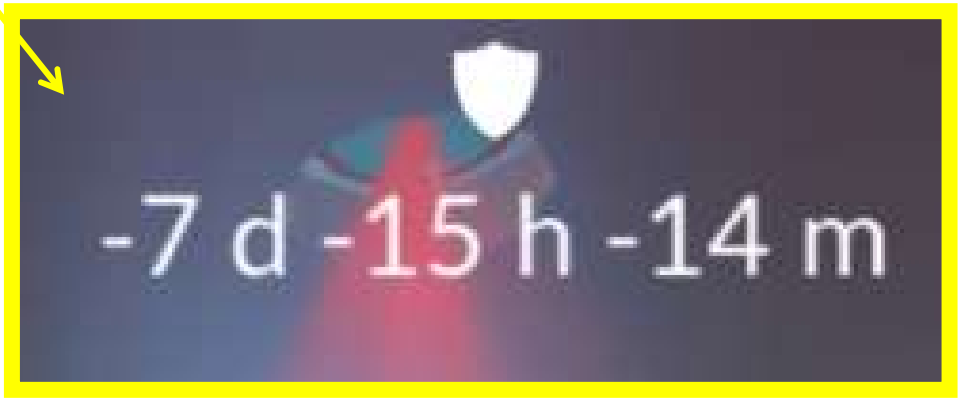
```
   01111111
 + 00000001
------------
   10000000 (二進位，最左邊代表正負號)
```

  - 對應到十進位是-128，而不是+128
  - +128已經超過8位元能表達的範圍，此時就發生”溢位”現象。

溢位 (overflow)

- 通常可以依據下列原則判斷是否發生溢位現象：
  - 當運算結果超出表達範圍
    (要先知道機器使用幾個位元儲存數值，如32 or 64 bit)。
  - 運算結果的正負符號出現異常狀況，例如：正整數+正整數的答案應該也是正整數，而運算結果的顯示為負數時，就代表發生了溢位現象。

掉進時空隙縫的寶可夢？



看見奇怪的負值，第一個可以懷疑的原因就是溢位問題...

（軟體工程師，你累了嗎？）