ABAP DEVELOPMENT

ABAP DEVELOPMENT

SAP ABAP 程式設計基礎課程
第三堂課：數據輸出及流程控制

歡迎您成為ABAP程式開發者,在本課程中,我們將瞭解到:

如何使用WRITE語句將數據顯示在螢幕及程式流程控制的相關語句

製作：周台誠
noah.chou@outlook.com

- 在**ABAP/4** 中常會使用**WRITE** 語句在螢幕或是報表紙上，產生複雜的輸出報表 。本章節所說明的內容也是之後自定報表的基礎。
  語法：**WRITE <f>.**
  - **<f>** 數據項目，可以下列型態：
    - 任何資料對象
    - 欄位符號或公式參數
    - 文本符號
  - 在螢幕上，如果同時使用多個 **WRITE** 語句，輸出欄位就一個接一個顯示，輸出之間由列分開（ 如一個空格 ）。如果當 前行沒有足夠空間，則開始新行。
  - 如果同時輸出多項，必須加冒號「,」分隔。

```
1  WRITE: 'COMPANY:', STFL-CARRID.
2  *使用冒號「,」來分隔多個輸出項目
```

- 預設資料類型的輸出格式

資料類型	輸出長度	定位
C	欄位長度	左對齊
D	8	左對齊
F	22	右對齊
I	11	右對齊
N	欄位 長度	左對齊
P	2 * 欄位長度（ +1）	右對齊
T	6	左對齊
X	2 * 欄位長度	左對齊

- 數字數據類型 **F**、**I** 和 **P** 是右對齊的，左邊用空格填充。如果有足夠的空間，也輸出千位分隔符。如果類型 **P** 欄位包含小數位，則預設輸出長度增加一位。

- 使用下列語句；可以在螢幕上定位 **WRITE** 語句的輸出。
  語法：**WRITE AT [/][<pos>][(<len>)] <f>.**
  - 斜線 '/' 表示新的一 行
  - **<pos>** 是最長為三位元數字的數字或變數， 表示在螢幕上的位置
  - **<len>** 是最長為三位元數字的數字或變數， 表示輸出長度

```
1  WRITE 'First line.'.
2  WRITE 'Still first line.'
3  WRITE /   'Second line.'
4  WRITE /13 'Third line.'
5  *如果輸出只包含直接值（即，不是變數），可以忽略關鍵字 AT。
```

```
First line. Still first line.
Second line.
            Third line.
```

```
1  DATA: LEN TYPE I VALUE 10,
2        POS TYPE I VALUE 11,
3        TEXT(10)   VALUE '1234567890'.
4  WRITE 'The text ------------ appears in the text.'.
5  WRITE AT POS(LEN) TEXT.
6  *因為第二個WRITE語句有指定輸出位置。因此覆蓋到上一個輸出的內容。
```

```
The text -1234567890- appears in the text.
```

```
1  DATA: NUMBER TYPE I    VALUE '1234567890',
2                TEXT(10) VALUE 'ABCDEFGHIJ'.
3
4  WRITE: (5) NUMBER, /(6) TEXT.
5  *如果輸出長度<len>太短，則只會顯示幾個字元。
6  *數字欄位會於左邊截斷，並用星號（ *）作首碼 。
7  *其它所有字段會於右邊截斷，但不會顯示出該字段較短的指示。
```

```
*7890
ABCDEF
```

- **WRITE** 語句，可以使用不同的格式化選項，來指定輸出格式
  語法：**WRITE .... <f> <選項>.**
- 所有資料類型的格式化選項

選項	用途
LEFT-JUSTIFIED	輸出左對齊。
CENTERED	輸出居中。
RIGHT-JUSTIFIED	輸出右對齊。
UNDER <g>	輸出直接開始於欄位 <g> 下。
NO-GAP	忽略欄位 <f> 後的空格。
USING EDIT MASK <m>	指定格式範本 <m>。
USING NO EDIT MASK	撤消對 ABAP/4 詞典中指定 的格式範本 的啟動。
NO-ZERO	如果欄位僅包含零，則用空格代替它們。對類型 C 和 N 欄位，將自動代替前導零。

- 日期欄位的格式化選項

選項	用途
DD/MM/YY	用戶主記錄中定義的分隔 符
MM/DD/YY	用戶主記錄中定義的分隔 符
DD/MM/YYYY	用戶主記錄中定義的分隔 符
MM/DD/YYYY	用戶主記錄中定義的分隔 符
DDMMYY	無分隔符號。
MMDDYY	無分隔符號。
YYMMDD	無分隔符號。

```
1  WRITE: SY-DATUM,
2         / SY-DATUM YYMMDD.
3  *將系統日期更改為無分隔符號
```

```
2013/07/04
130704
```

- 數字欄位的格式化選項

選項	用途
NO-SIGN	不輸出前置字元號。
DECIMALS <d>	<d> 定義小數點後的數字位元數。
EXPONENT <e>	在類型 F 欄位中，在 <e> 中定義冪數 。
ROUND <r>	用10**(-r) 乘類型P 欄位，然後取整。
CURRENCY <c>	按表格 TCURX 中的貨幣 <c> 格式化。
UNIT <u>	按表格 T006 中為類型 P 欄位所指定的單位 <u> 固定小數位數。

```
5  DATA  FLOAT TYPE F VALUE '123456789.0'.
6  WRITE FLOAT EXPONENT 3.
7  *將資料類型為 F 的變數定義冪數
```

```
123456.78900000000E+03
```

```
1  DATA: G(5) VALUE 'Hello',
2        F(5) VALUE 'Dolly'.
3  WRITE: G, F.
4  WRITE: /10 G,
5         /   F UNDER G.
6  *直接輸出於 變數 G 下方。
7  WRITE: / G NO-GAP, F.
8  *忽略變數 G 後的空格
```

```
Hello Dolly
          Hello
          Dolly
HelloDolly
```

```
10 DATA   TIME TYPE T VALUE '154633'.
11 WRITE: TIME,
12   /(8) TIME USING EDIT MASK '__:__:__'.
13 *指定格式範本為：__:__:__
```

```
154633
15:46:33
```

```
15 WRITE: '000123',
16      / '000123' NO-ZERO.
17 *如果欄位僅包含零，則用空格代替它們。
18 *對類型 C 和 N 欄位，將自動代替前導零。
```

```
000123
   123
```

- 可以顯示系統中所提供的符號或圖示
  語法:
  WRITE <symbol-name> AS SYMBOL
  WRITE <icon-name> AS ICON

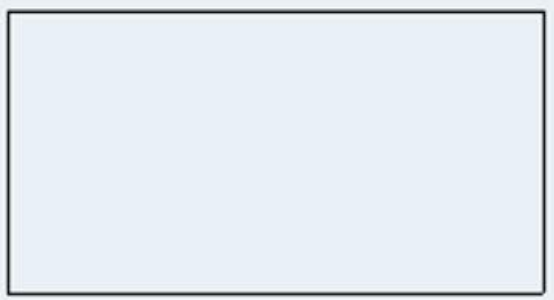
```
1  INCLUDE <SYMBOL>.
2  INCLUDE <ICON>.
3  WRITE: /  'Phone Symbol:', SYM_PHONE AS SYMBOL.
4  WRITE: /  'Alarm Icon:  ', ICON ALARM AS ICON.
```

```
Phone Symbol: ☎
Alarm Icon:
```

- 如果要查看系統所提供有哪些符號及圖示，可選擇返回**ABAP**編輯器的初始頁，執行程式 **SHOWSYMB**和**SHOWICON** 就會列出所有符號和圖標

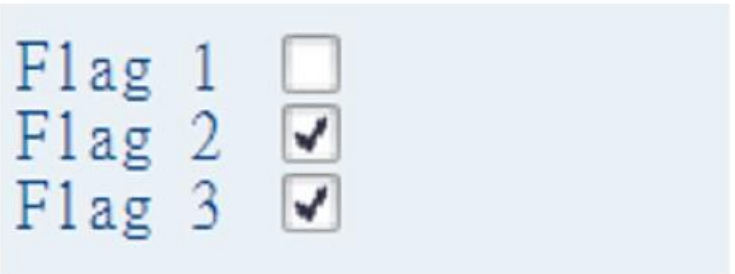
- 產生水平線
  語法： **ULINE [AT [/][<pos>][(<len>)]].**
  WRITE [AT [/][<pos>][(<len>)]] SY-ULINE.
  WRITE [AT [/][<pos>][(<len>)]] '------'.
- 產生垂直線
  語法： **WRITE [AT [/][<pos>]] SY-VLINE.**
  WRITE [AT [/][<pos>]] '|'.
- 產生空白行
  語法： **SKIP [<n>].**
- 跳至指定列坐标
  語法： **SKIP TO LINE [<n>]**

```
 6  WRITE 5(20) SY-ULINE.
 7  WRITE /5 '|                                  |'.
 8  WRITE /5 '|                                  |'.
 9  WRITE /5 '|                                  |'.
10  WRITE /5 '|                                  |'.
11  WRITE /5(20) SY-ULINE.
```



- 使用下列語法，可以將欄位的第一個字元，作為核取方塊輸出到螢幕上：
  語法： **WRITE <f> AS CHECKBOX.**
  如果欄位 **<f>** 的第一個字符是一個“ **X**”，就顯示核取方塊已選取。如果欄位 **<f>** 的第一個字符是 **SPACE**，就顯示複選框為未選取。

```
1  DATA: FLAG1    VALUE ' ',
2        FLAG2    VALUE 'X',
3        FLAG3(5) VALUE 'Xenon'.
4  WRITE: / 'Flag 1 ', FLAG1  AS CHECKBOX,
5         / 'Flag 2 ', FLAG2  AS CHECKBOX,
6         / 'Flag 3 ', FLAG3  AS CHECKBOX.
7  *FLAG2 和 FLAG3， 因為這些字段的第一個字元是"X"
8  *所以核取方塊為已選取。按一下滑鼠就可以改變核取方塊的狀態。
```



運算子	含義
EQ 或 =	等於
<> 或 >< 或 NE	不等於
< 或 LT	小於
<= 或 LE	小於等於
> 或 GT	大於
>= 或 GE	大於等於
AND	且
OR	或
NOT	反

- 如果要要比較字串（類型 **C**）和數位文本（類型 **N**），可以在邏輯表達式中使用下列運算子

運算子	含義
CO	僅包含
CN	不僅包含
CA	包含任何
NA	不包含任何
CS	包含字串
NS	不包含字串
CP	包含模式
NP	不包含模式

- 要檢查值是否屬於特定範圍，請使用下列帶有**BETWEEN**參數的邏輯運算式：
  語法：**<F1> BETWEEN <F2> AND <F3>**
  如果**<F1>**在**<F2>**和**<F3>**之間的範圍內發生，則運算式為真，下列為邏輯運算式的短格式：
  IF <F1> GE <F2> AND <F1> LE <F3>.

- 要檢查欄位是否設置為初始值，請如下使用帶有**IS INITIAL**參數的邏輯運算式。
  語法： **<f> IS INITIAL**
  如果**<f>**包含本身類型的初始值，則運算式為真。通常情況下，任何欄位，基本的或結構化的（字元串和內表），在**CLEAR <f>**語句執行後，都包含其初始值。

- 檢查欄位內容是否與選擇表中的選擇條件匹配，可以如下使用帶有**IN**參數的邏輯運算式：
  語法： **<f> IN <seltab>**
  如果欄位**<f>**內容的符合選擇表**<seltab>**中的條件，則運算式為真。

- **IF 的條件語句**

語法:

```
IF <條件1>.
   <陳述式區塊1>
ELSEIF <條件2>.
   <陳述式區塊2>
ELSEIF <條件3>.
   <陳述式區塊3>
.....
ELSE.
<else 陳述式區塊>
ENDIF.
```

```
1  DATA: TEXT1(30) VALUE 'This is the first text',
2        TEXT2(30) VALUE 'This is the second text',
3        TEXT3(30) VALUE 'This is the third text',
4        STRING(5) VALUE 'eco'.
5  IF      TEXT1 CS STRING.
6     WRITE / 'Condition 1 is fulfilled'.
7  ELSEIF TEXT2 CS STRING.
8     WRITE / 'Condition 2 is fulfilled'.
9  ELSEIF TEXT3 CS STRING.
10    WRITE / 'Condition 3 is fulfilled'.
11 ELSE.
12    WRITE / 'No condition is fulfilled'.
13 ENDIF.
14 *這裡，第二個邏輯表達式TEXT2 CS STRING為真
15 *因為字串"eco"存在於TEXT2中。
```

- 在每個判斷語句之後要加上句點「.」
- 在巢狀循環（循環嵌套）之中無法使用 **ELSE** 語句，**ELSE** 語句屬 **IF** 語句

- **CASE的條件語句**

語法:

CASE <變數f>.
WHEN <值1>.
<陳述式區塊1>
WHEN <值2>.
<陳述式區塊2>
....
WHEN OTHERS.
<others陳述式區塊>
ENDCASE.

```
1  DATA: TEXT1    VALUE 'X',
2        TEXT2    VALUE 'Y',
3        TEXT3    VALUE 'Z',
4        STRING   VALUE 'A'.
5  CASE STRING.
6    WHEN TEXT1.
7      WRITE: / 'String is', TEXT1.
8    WHEN TEXT2.
9      WRITE: / 'String is', TEXT2.
10   WHEN TEXT3.
11     WRITE: / 'String is', TEXT3.
12   WHEN OTHERS.
13     WRITE: / 'String is not', TEXT1, TEXT2, TEXT3.
14 ENDCASE.
15 *因為STRING 的內容"A" 不等於"X" 、"Y"或 "Z"
16 *所以會執行 WHEN OTHERS 後面的語句區塊
```

- 如果想要多次執行陳述式區塊，則可以使用**DO**語句來設計程式迴圈
  語法：
  DO [n TIMES] [VARYING <f> FROM <start> NEXT <end> [RANGE <range>]].
  <陳述式區塊>
  ENDDO.
- 在發現**EXIT**、**STOP**或**REJECT**語句之前，系統將繼續執行由**DO**引導、**ENDDO**結束的語句區塊。
- 可以使用**TIMES**選項限制循環次數。**n** 可以是文字或變數。如果**<n>**是**0**或負數，系統不執行該迴圈。
- 系統欄位**SY-INDEX**中包含已處理過的迴圈次數。

```
1  DO.
2  WRITE SY-INDEX.
3     IF SY-INDEX = 3.
4        EXIT.
5     ENDIF.
6  ENDDO.
7  *DO迴圈的基本格式，處理3次迴圈，
8  *然後在EXIT語句後退出迴圈。
```

- 可以任意嵌套**DO**迴圈，也可以與其他循環組合使用

```
 9  DO 2 TIMES.
10     WRITE SY-INDEX.
11     SKIP.
12     DO 3 TIMES.
13         WRITE SY-INDEX.
14     ENDDO.
15     SKIP.
16  ENDDO.
17  *外部迴圈執行2次。每次執行外部迴圈時，內部循環都執行3次。
18  *注意系統欄位SY-INDEX記錄每個循環各自的循環次數。
```

- VARYING選項在每次迴圈中給變量<f>重新賦值。<F1>、<F2>、<F3>、...必需是記憶體中類型相同和長度相等的一系列等距欄位。第一次迴圈中，將<F1>分配給<f>，第二次迴圈中，將<F2>分配給<f>，以此類推。可以在一個DO語句中使用多個VARYING選項。
- 需保證迴圈次數不超過涉及到的變量<F1>、<F2>、<F3>的數量。

```
DATA: BEGIN OF TEXT,
      WORD1(4) VALUE 'This',
      WORD2(4) VALUE 'is',
      WORD3(4) VALUE 'a',
      WORD4(4) VALUE 'loop',
      END   OF TEXT.
DATA: STRING1(4), STRING2(4).
DO 4 TIMES VARYING STRING1 FROM TEXT-WORD1 NEXT TEXT-WORD2.
     WRITE STRING1.
   IF STRING1 = 'is'.
      STRING1 = 'was'.
   ENDIF.
ENDDO.
SKIP.
DO 2 TIMES VARYING STRING1 FROM TEXT-WORD1 NEXT TEXT-WORD3
           VARYING STRING2 FROM TEXT-WORD2 NEXT TEXT-WORD4.
     WRITE: STRING1, STRING2.
ENDDO.
*欄位串TEXT代表記憶體中四個等距字段序列。
*每次執行第一個DO迴圈時，都依次將其組件分配到STRING1中。
*如果STRING1包含"is"，則將其改變為"was"，而且自動將TEXT-WORD2改變為"was"。
*每次執行第二個DO迴圈時，將TEXT的組件傳遞給STRING1和STRING2。
```

- 如果只要條件為真，就不止一次執行語句，可以如下使用**WHILE**語句來設計程式迴圈
  語法:
  WHILE <條件> [**VARY** <f> **FROM** <F1> **NEXT** <F2>].
  <陳述式區塊>
  ENDWHILE.
- 只要<條件>是真，或系統發現**EXIT**、**STOP**或**REJECT**語句，系統將繼續執行由**WHILE**語句引導、**ENDWHILE**結束的語句塊。
- 對於<條件>，可以使用程式設計邏輯表達式中描述的任何邏輯表達式。
- 系統欄位**SY-INDEX**中包含已執行的迴圈次數。
- **VARY**選項與**DO**迴圈的**VARYING**選項工作方式一樣。允許每次執行循環時為變數<f>重新賦值。<F1>、<F2>、<F3>、...必需是記憶體中類型相同和長度相等的一系列等距欄位。第一次迴圈時，將<F1>分配給<f>，第二次迴圈時，將<F2>分配給<f>，以此類推。可以在一個**WHILE**語句中使用多個**VARY**選項。

```
1  DATA:    LENGTH      TYPE I VALUE 0,
2           STRL        TYPE I VALUE 0,
3           STRING(30) TYPE C VALUE 'Test String'.
4  STRL = STRLEN( STRING ).
5  WHILE STRING NE SPACE.
6     WRITE STRING(1).
7     LENGTH = SY-INDEX.
8     SHIFT STRING.
9  ENDWHILE.
10 WRITE: / 'STRLEN:                ', STRL.
11 WRITE: / 'Length of string:', LENGTH.
12 *此處使用WHILE迴圈確定字符串的長度。
13 *做法是：每次執行循環時，都將字串左移一位，直到僅包含空格。
14 *確定字元串長度更簡便和有效的辦法是通過使用內置的函式STRLEN。
```

```
T e s t   S t r i n g
STRLEN:                    11
Length of string:          11
```

- **CONTINUE** 跳至下一次循環

```
1  DO 3 TIMES.
2       IF SY-INDEX = 2.
3          CONTINUE.
4       ENDIF.
5       WRITE / SY-INDEX.
6  ENDDO.
```

- **CHECK** <條件> **CHECK** 之後條件成立才繼續往下執行循環

```
7  DO 5 TIMES.
8       CHECK SY-INDEX BETWEEN 2 AND 4.
9       WRITE / SY-INDEX.
10 ENDDO.
```

- **EXIT** 跳離循環

```
11 DO 10 TIMES.
12      IF SY-INDEX = 4.
13         EXIT.
14      ENDIF.
15      WRITE / SY-INDEX.
16 ENDDO.
```

- LOOP語句通常用於實現內表数據的循環讀取及操作。
  語法：LOOP
  <陳述式區塊>
  ENDLOOP

```
1  LOOP AT ITAB.
2    WRITE: ITAB.
3  ENDLOOP.
```