電子計算機科學叢書

郭德盛　主編

福傳程式設計技巧

增訂版

陳　秋　發

編　著

田野出版社　印行

福傳程式設計技巧

陳

田野

福傳程式設計技巧

編著

陳秋發

國立台灣大學電機工程學系副教授

田野出版社印行

福傳程式設計技巧

書號：210203

每本定價新台幣 110 元正

編著者：陳　秋　發
發行人：陳　碧　玉
發行所：田野出版社
台北市仁愛路二段一一〇號三樓
電話：3930255 · 3930249
總經銷：松崗電腦圖書資料有限公司
台北市仁愛路二段一一〇號三樓
電話：3930255 · 3930249
經　銷：文　笙　書　局（TEL：3810359）
承印者：泉崗印刷設計股份有限公司
台北市仁愛路二段一一〇號三樓
電話：3930255 · 3930249

中華民國六十四年 五 月　初 版
中華民國七十一年 九 月　第六版
中華民國七十二年 五 月　增訂版

本出版社經行政院新聞局核准登記，
登記證為局版台業字第〇二〇七號。

電子計算機科學叢書

郭德盛博士主編

1. 電子計算機名詞字典，許照校訂，郭德盛主編
2. 電子計算機程式語言——COBOL，吳建平編譯
3. 數字方法，陳秋發編著
4. 電子計算機原理，郭德盛編著
5. 系統分析與設計，鍾英明編著
6. COBOL 程式設計範例，鍾英明編著，吳建平校閱
7. 電子資料處理，吳建平編著
8. 編傳程式設計技巧，陳秋發編著
9. 編傳程式設計，李學養編著
10. 微計算機基本原理，子惠中編譯
11. 商用程式語言 COBOL，鍾英明、陳秋發合著
12. 計算機科學概論（上册），李學養編譯
13. 計算機科學概論（下册），李學養編譯
14. 資料結構，劉乃誠、吳建平合著
15. 培基程式語言 BASIC，子惠中、周曜榮合著
16. 程式語言 FORTRAN 77，郭德盛、許舜欽合著

序

本書之目的，在於介紹一些常用的福傳程式設計技巧。這些技巧對專家而言，可能都已十分熟悉，但對於祇學過程式設計語言，或想在這方面多培養一些能力的人而言，是很有幫助的。目前，在工程及科學研究方面，福傳仍是最合適且使用最廣的計算機語言，也是大專理工科學生必修課程之一。現在有關福傳語言的書籍之多，已達令人驚訝之地步。但多數爲語言之介紹，或是有關數學問題之解法的探討，而非數值之應用的技巧描述則非常少，編者有鑑於此，乃撰寫此書，期對學習程式設計者有所幫助。

本書着重於實用技巧之介紹，大部份內容都不涉及深奧的數學，可以說是適用於各方面的。在每種技巧提出時，都附有適當的福傳敍述實例說明的文句，使讀者易於了解和知道如何應用。其中有些比較特殊的技巧，可能並不是人人都用得着的，則僅約略說明一下，細節方面留待讀者自行寫出。

本書承李學養教授之校閱，及多位同仁之熱心協助，謹此致謝。本書之編著係利用課餘之暇，漏誤之處，在所難免，尚祈各先進不吝指正。

陳　秋　發　民國六十四年五月

於國立台灣大學

目　錄

序

第一章　基本技巧

1-1　旗標與開關……1
1-2　ＤＯ迴圈……3
1-3　數字之包裝……4
1-4　數字之解開……5
1-5　編　　表……6
1-6　緩衝器……8
1-7　開放式次常式……9
1-8　字元……11
1-9　綫性搜尋……13
　　習　題……16

第二章　數字與字元

2-1　字元變整數的轉換……19
2-2　整數變字元的轉換……21
2-3　字元變浮點數的轉換……25
　　習　題……26

第三章　利用列印機繪圖

3-1　基本概念……27
3-2　帶狀圖……28

3-3 點　圖…… 32
3-4 綫形圖…… 35
3-5 密度圖…… 38
習　題…… 45

第四章　搜　　尋

4-1 二分搜尋…… 47
4-2 用綫性搜尋的混雜法…… 49
4-3 用非綫性搜尋的混雜法…… 52
習　題…… 55

第五章　字元和字

5-1 字元的識別…… 57
5-2 關鍵字的識別…… 59
5-3 字的識別…… 61
習　題…… 64

第六章　儲存箱和佇列

6-1 儲存箱…… 65
6-2 邏輯…… 66
6-3 佇列…… 73
6-4 雙頭佇列…… 75
習題…… 77

第七章　串列處理

7-1 鏈式串列…… 81
7-2 雙重鏈式串列…… 85

7-3 樹型串列…… 87
習 題…… 89

第八章 分 類 法

8-1 交換分類法…… 91
8-2 波狀分類法…… 93
8-3 競比分類法…… 96
8-4 串列處理分類法…… 100
習 題…… 107

第九章 符號狀態表

9-1 簡單語法核對…… 111
9-2 動作之調用…… 114
9-3 次常式之調用…… 118
9-4 表之自動建立…… 120
習 題…… 122

第十章 結構化程式設計

10-1 何謂結構化程式設計…… 125
10-2 由上而下的設計…… 126
10-3 程式的結構…… 130
10-4 結構化之注意事項…… 135
習 題…… 137

附錄一 程式設計之一般原則…… 139

附錄二 一般常犯的錯誤…… 163

附錄三　程式範例 …… 169
參考書目 …… 193
索　　引 …… 195

第一章　基本技巧

在這一章裡面，我們首先介紹一些基本的程式設計技巧，它們可看作是基本的單位，用以構成較複雜的技巧。以後幾章內，這些基本的技巧會一再地被應用到。

1-1　旗標與開關

有時候我們需要利用**旗標**（Flag）或**開關**（Switch）來記錄計算過程中的某一些現狀。這種開關可以被打開或關掉，其值可用以測試種種狀態。

要達到上述的目的，我們可以使用一個**邏輯變數**（Logical variable），設爲LFLG。這個變數的值不是眞（.TRUE.）就是假（.FALSE.）。利用下面的敍述，可以把這個開關由這個值，變到另一個值。

```
LFLG = .NOT.LFLG
```

這個值可以用**邏輯IF**（Logical IF）敍述來測試：

```
IF (LFLG)  GO TO 12
```

上述的控制方式也可以利用**整變數**（Integer variable）來達成，設該整變數爲IFLG。我們使用1與2兩個值來代表兩種不同的狀態。這個開關可以利用下面的敍述，從一個值變成另一個值（即從1變爲2或由2變爲1）：

```
IFLG = 3 - IFLG
```

然後利用**計值GO TO敍述**（Computed GO TO statement），來測試這個值，以控制轉到那一個敍述去

```
GO TO (12,22), IFLG
```

有時候，所需要的一種開關要能夠循環地指定連續的值才可以。例如，我們所要的開關，將取的值爲1，2，3，4，5，1，2……等等。這可以利用下面的敍述，使這個開關的值變到下一個值：

```
IFLG = IFLG + 1
IF ( IFLG.GT.5 )  IFLG = 1
```

也可以只用一個敍述來完成相同的工作：

```
IFLG = MOD ( IFLG,5 ) + 1
```

MOD 是一個非常有用的**內在函數**（Intrisic function）。MOD（I，J）的結果是I除以J的餘數。例如，IFLG = 3時，MOD（IFLG，5）= 3。然而，此時利用MOD 函數的效率卻比較低，雖然用**福傳**（FORTRAN）寫出的程式較短，但大多數的情況下，機器執行這個敍述時，比執行上面的兩個敍述要作更多的工作。這種開關的值也可以用計值GO TO敍述來測定：

```
GO TO ( 12 , 22 , 32 , 2 , 15 ) , IFLG
```

如果這個開關所取的值是依照相反的方向，即5，4，3，2，1，5，4，3……等等，則可以利用下面的敍述使它的值變化到下一個值：

```
IFLG = IFLG - 1
```

```
IF ( IFLG.LT. 1 )  IFLG = 5
```

1-2　DO 迴圈

福傳的**DO迴圈**（ DO loop ），在計算機內可以非常有效率地執行，但這種效率是一些設定的限制所付出的代價。它的**指標變數**（ Index variable ）必須是整變數，就很不方便；其次它的**初值**（ Initial value ），**終值**（ Final value），與**增值**（ Increment ），必須爲正數，更顯得不便。

通常不直接適用DO 敍述的迴圈，可以利用下面兩種方法達成：

(1)程式師（ Programmer ）可以用一些**指定敍述**（ Assignment statement ），定下初值，增值及終值等，再配合上適當的 IF 和GO TO 敍述即構成迴圈，但這種處理方式的效率較低，因它所要的執行時間較多。

(2)可以利用通常的DO 迴圈，多用一個或更多的指定敍述於此迴圈內，來建立所要的數值。此時，在迴圈末了的增值及測試的效率仍然保留。

假如一個迴圈要求 J 的值由－43變到＋43（即增值爲 1 ），則由第二種方法來處理時，所需的敍述如下：

```
DO  15   I = 1 , 87
J = I - 44
```

假如有一個迴圈要求 J 值有連續的偶數值，從 16 到 4 （即增值爲－2 ），則所需的敍述爲：

```
DO  25   I = 4 , 16 , 2
J = 20 - I
```

設有個廻圈要一個實數指標A，取連續的值0.1，0.2，0.3…，1.2。這可以利用下面的敍述加以解決，即

```
DO  35   I = 1 , 12
A = FLOAT ( I ) / 10.0
```

這個問題，也可以利用設A的初值爲0.0，而每循環一次就增加0.1的方法。但這樣不太好，因爲0.1在計算機內化成二進制時，並不是完全等於0.1，而是有誤差的，所以誤差將隨每次A值的增加而增加。

有時多於一個的指定敍述是需要的。假設需要一廻圈，其中N值取1、2、4、8、16、32、64、128、256、512……等等。換句話說，N每次增值一倍，解決之法爲：

```
      N = 1
      DO  45   I = 1 , 10
      ..................
45    N = N + N
```

此外，只用一個指定敍述也可以達到這個目的：

```
      DO  45   I = 1 , 10
45    N = 2 ** ( I - 1 )
```

然而，此時每循環一次就要執行一次指數運算，而前一種方法只要一次加法運算。注意：N自加一次比每次N乘2要來得有效率，這是因爲加法運算所需的時間通常比乘法要來得少，所以較快。

1-3 數字之包裝

爲了節省儲存位置，一些小的整數數字可以包裝（Packing）起來，放入一個整變數內。不過這些數字不可以是負數。如果這些整數的

值在0與n之間，這個方法是把它們看成**底數**（Base）爲n＋1的連續數位。我們首先把第一個整數放入這變數，乘以n＋1後把第二個整數加入這變數，如此繼續下去。

例如，有個**向量**（Vector）IDIG，有五個元素，其值分別爲6，8，0，3，1，這些可以包裝起來放入一個變數NUM，作爲一個十位數的數字（此處假定一個整變數所能存放的數值高達有效位數十位）。方法如下：

```
      NUM = 0
      DO  55   I = 1 , 5
55    NUM = NUM *  10 + IDIG ( I )
```

執行完這些敍述後，NUM將包含值68031。同樣一串二進數字（只含0和1）也可以放入一個整變數。只要在加上一個新數值之前，這變數先乘以2即可。

在包裝的過程中所產生的數字之最大值，絕對不可超過計算機所允許的最大正整數值。

1-4　數字之解開

假如一些很小的整數，已經利用上述的方法包裝在一個變數內，則可以利用MOD函數來**解開**（Unpacking）。第一次除以n＋1的餘數，所得的數字也就是最後被包裝的數字。如此繼續下去，就可以得到所有被包裝的每一個數字。

利用上一節的例子，如果NUM含有值68031，則利用下列敍述，可以將這些連續的十進位數字，分別放入向量IDIG的五個元素內，

```
      DO  65   I = 1 , 5
      J = 6 - I
```

```
      IDIG(J)=MOD(NUM, 10)
65    NUM=NUM/10
```

實際上這是一個增值爲－1的DO 迴圈。因爲第一個從NUM 解開的數字，必須送入IDIG的第五個元素。

同樣地，二進位數值也可以從被包裝的變數解開出來，祇要上列敍述內用2取代10 即可。顯然我們也可以利用MOD（NUM，2）測試NUM 是奇數或是偶數。假如結果是0則NUM 是偶數，假如結果是1則NUM 是奇數。此外，NUM的第M＋1個位元（Bit ）可以利用下面的**陳式**（Expression）來解開：

```
MOD(NUM/(2**M),2)
```

例如，NUM 的第7個位元可以利用下面的敍述設爲1（無論它原來的值是什麼）：

```
IF (MOD(NUM/64,2).NE.1) NUM=NUM+64
```

類似的敍述則可以使某個位元定爲0（無論它原來的值是什麼）。在福傳語言內，有相當數量的位元處理是可以如此執行的，但是却非常無效率。最後要提醒注意的是，在所有的位元處理中，整變數不得變爲負數。

1-5 編　　表

有時候會遇到一組數字需要不規則地映至另一組的數字，以致沒有辦法用計算的方法來完成。如果第一組的數字是整數，則可以將它們當作**註標**（Subscript ），用來表示另一組數字儲存在向量內的位置。

假定MONTH 之整數值在1與12 之間，以代表一年中的某個月份，NDAYS 則爲那個月份的日數（不管閏年即二月以28 日計）。假設向量LMONTH有12個元素，可經由下面的敍述給定好初值：

```
      DATA  LMONTH／31,28,31,30,31,30,31,31,30,31,30,
     *                    31／
```

此時，適當的日數會經由下面敘述而放入NDAYS內：

```
      NDAYS = LMONTH（MONTH）
```

這個方法比用12 個邏輯IF 敘述或有12 個分支的計值GO TO敘述都來得有效和簡便。

假設需要一個迴圈，它的指標取如下的連續值1.5,2.0,2.5,…直到4.0（每次增加0.5），然後5.25,6.5,7.75，……直到11.5（每次增加1.25），然後13.0,14.5,16.0，最後爲20.0。如此一來，就需要5個指標的增值爲0.5，6個指標的增值爲1.25，3個指標的增值爲1.5，及一個指標的增值爲4.0 等增量不同的變化。這個問題用一般的DO 迴圈處理將十分不便，但用這裡所說的方式却可以很優美地來完成它，只要把這些增值存入一個向量內，而這些增值所使用的次數存於另一向量內。每次需要一個新的增值時，它的值及使用次數，就從表中挑出來使用。這指標所假設的連續值的數目，比所需的增值的數目多1。這是指如果這個指標是在迴圈的末了才增加的話，它必須多執行一次，使這個迴圈能正確地工作。結果如下：

```
      DIMENSION AINC（4），NINC（4）
      DATA AINC／0.5,1.25,1.5,4.0／,NINC／5,6,3,2／
      A = 1.5
      DO 75  I = 1，4
      J = NINC（I）
      ZINC = AINC（I）
      DO 85  K = 1，J
      ⋮
```

```
85   A = A + ZINC
75   CONTINUE
```

1-6 緩衝器

程式常常經由分散的計算，一次只能產生一個結果。如果這些結果在它們產生時就要立刻印出，則每行只印一個數，每一福傳的WRITE敘述就要再從新的錄（Record）開始。這樣的話，有時會覺得太浪費或印出的形式不是所希望的。解決的方法，就是把這些結果先存於一個向量內，在裝滿之後再把它們印出來。這個向量就稱之爲**緩衝器**（Buffer）。

在下面的例子中，向量BUFF 當作緩衝器，設有十個元素。NBUFF 表示目前存在BUFF中的數值之個數，開始時定爲 0。新產生的值則存於VALUE 內，下列的敘述，可以把新的結果陸續存入緩衝器內，如果緩衝器滿了，就把它們印出來：

```
      NBUFF = NBUFF + 1
      BUFF ( NBUFF ) = VALUE
      IF ( NBUFF  .LT. 10 )  GO TO 50
      WRITE ( 6 , 300 )  BUFF
300   FORMAT ( 1X , 10F12.4 )
      NBUFF = 0
 50   CONTINUE
```

如果程式沒有產生剛好十的倍數的運算結果，則在程式結束以前，要把緩衝器內所存的部份結果印出來。可以用下列的敘述來完成：

```
  IF (NBUFF.GT.O)  WRITE ( 6 , 300 )  (BUFF ( I ),
*I = 1 , NBUFF )
```