高等結構化 COBOL程式設計

李惠明　編著

萬人出版社出版

高等結構化COBOL程式設計

萬館卡電字第81—1007號

編著者：李　惠　明
出版者：萬人出版社有限公司
發行人：呂　芳　烈
電　話：3144895・3819425
地　址：台北市重慶南路一段41號
郵　撥：119410 萬人出版社帳戶
出版證：局版台業字第 1882 號
印刷者：啟盟印刷廠股份有限公司
台北市長泰街121巷10弄2號
中華民國七十一年三月　初版
中華民國七十一年十二月　三版

特價 200 元

序

電子計算機的迅速發展，使得它的應用亦日趨普及；近年來由於政府機關及公民營企業的普遍使用電子計算機，使得電子資料處理人員的需求亦隨而增加。COBOL 程式語言一直是電子資料處理的主要通用語言，使用最爲廣泛者之一。

本書爲著者前著「結構化 COBOL程式設計」一書的續著，編寫要旨如下：

1.內容詳盡：對於各種特殊指令的用法，作詳細的解說，期使讀者能更深入了解COBOL 程式語言的應用方法。

2.例題豐富：每章均有例題若干，期使讀者能了解實務的應用方法，俾理論與實務能相互配合。

3.習題豐富：經由習題的習作，當更能熟練精通。

著者期望本書讀者的是不僅能精通活用COBOL 程式語言，更能成爲一名出色的程式設計師。

本書之編著係利用公餘之暇，雖力求完美，唯著者才疏學淺，漏誤之處在所難免，尚祈各先進不吝指教。

李 憲 明 識

目　　錄

第一章　結構化程式設計法則

1-1　結構化程式設計法則的目標

以往的電腦程式時常遭到責難，例如不能迎合用戶的需求、無法準時完成、實際成本高於預估成本、含有錯誤、難以維護（爲迎合新的要求而修改）等。這些困難時常發生，因此有很多電腦機構曾經致力於改善程式設計的方法以期增進效能。而製訂出程式設計法則（Programming discipline），它是一種良好定義的常規、程序、以及開發控制過程。欲爲一名好的程式設計員，要學習該怎樣的設計方能成一個高品質的程式——它不僅容易瞭解、容易維護、而且該程式能夠很有效地運用電腦資源。

程式設計法則在軟體工業上非常重要，下面將說明這種程式設計方法的主要目標：

1. **迎合用戶的需要；**
2. **在預算範圍內準時完成；**

3. **沒有錯誤的程式敍令；**
4. **抵擋錯誤的作業；**
5. **維護性高的程式；**
6. **可攜帶的程式。**

1. **迎合用戶的需要：**

每一個程式各有其目的，諸如編製一份分析報告或計算一組統計數字，如果由於一個應用程式太繁雜或太難用，致使該程式的用戶不去用它，那麼這個程式的設計就是一種失敗。法則化的程式設計包含了在程式撰寫之前對用戶需求的小心分析。

2. **在預算範圍內準時完成：**

撰寫電腦程式所需時間和費用的估計往往發生低估現象。使用法則化的、結構化的方式來設計，很多電腦機構的生產力已經有了很大的改善，同時也改進了他們在預估完成的時間。

3. **沒有錯誤的程式敍令：**

一般都認爲所有的大型電腦程式都含有錯誤，而且無法把一個大型程式裏的每一項錯誤都除掉。事實上，使用法則

化的、結構化的方式來設計和開發程式，可以把錯誤的可能性減到最小，而且在預測階段很容易找出錯誤的地方而將它改正過來。其結果幾乎是一種無缺點的程式設計。

4. **抵擋錯誤的作業**：

一個程式有可能產生錯誤的結果，其原因不外乎程式錯誤或輸入資料不對。因此程式的設計必須要能夠在該程式的執行過程當中把可能的錯誤自己把它找出來，協助偵錯的設計特性包括：

(1) 輸入檢驗。

(2) 在處理過程中檢驗是否正確。

輸入檢驗是檢驗所有的輸入資料是否符合預定的條件。例如檢查輸入

① 資料是否含有必要的輸入資料項目；

② 資料項目的值是否在可接受的範圍之內；

③ 資料項目間是否符合一定的關係；

④ 資料項目的性質是否相符（數字項目或文字項目）。

處理過程中檢驗是否正確，一般說來可以檢查處理結果的合理性，以及各種結果之間的邏輯關係。

5. **維護性高的程式：**

電腦程式時常需要修訂，或由於條件的更改，或由於要改正錯誤。所以在撰寫程式時，要把該程式的維護工作考慮在內。程式的文書資料以及程式的型態應該要讓其他程式人員容易瞭解該程式的處理邏輯，而且在修改該程式的某一部份時不致於使得該程式的其他部份引入一項錯誤。

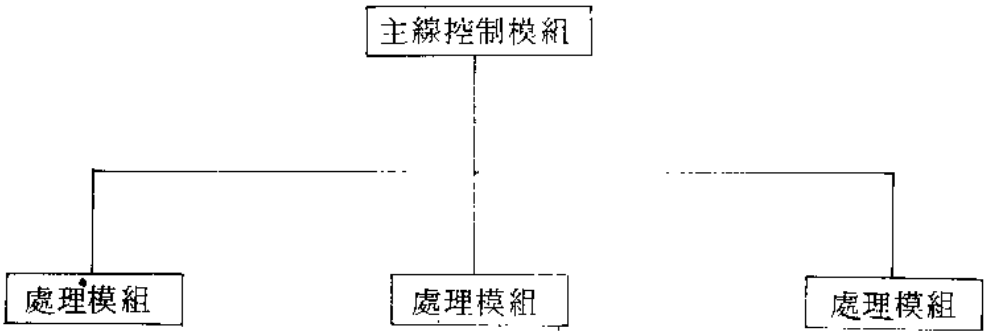
6. **可攜帶的程式：**

高階程式語言寫成的程式，應該不需經過太多的修改就可以在另一部有該種程式語言的編譯程式的電腦上執行。這就是說應該避免使用某一電腦獨具的特性或非標準的敍令。根據法則化的、結構化的方式所撰寫的直截了當、良好文書的程式是可攜帶的，而錯綜複雜、不良書寫邏輯的程式則否。

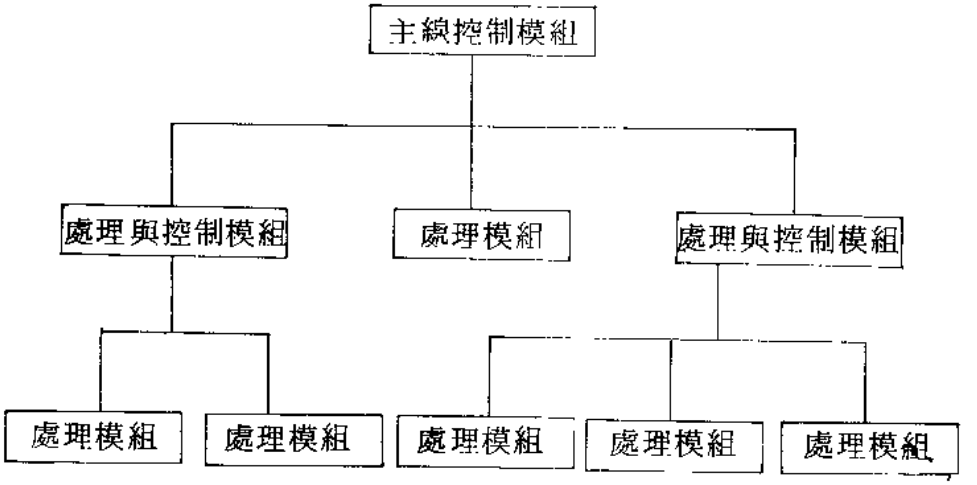
1-2 模組化程式設計

在應用程式設計法則上的一項重要觀念是設計一個由數個單元所組成的程式，這些單元叫做**模組**（Module ）。一個模組是一個程式的一部份，它執行一項單獨的功能如輸入、輸入檢驗、處理某一類的輸入資料等。一個程式模組如果相當大，包括相當複雜的邏輯，需要

相當多的敘令時，我們可以再細分爲數個次模組。這種再細分的過程一直繼續到每一個模組的複雜程度和敘令的個數降到可管理的範圍之內爲止。



(a)　簡單程式模組層次圖



(b)　三層的程式層次圖

圖 1-1　一個程式的模組結構

用模組化方法規劃的一個簡單電腦程式係由一個控制模組和一或多個處理模組所組成，控制模組導引處理模組的執行。一個較複雜的程式包含一個主線控制模組和一個處理模組的層次。主線控制模組導引第一層的處理模組，第一層的處理模組招喚其所屬的下一層模組。控制常規能夠用很簡單的方式來設計而無錯綜複雜的控制轉移，因之，程式的處理邏輯變成很顯明且易於閱讀，如此程式更易於維護和除錯。

功能模組	說明
主線控制	執行所有的判斷決定處理的順序和用以指示第一層處理模組的執行。
起始	執行程式作業準備處理步驟。
輸入	執行該程式所需資料的輸入工作。
輸入資料檢核	執行輸入資料的檢核以查知是否有錯誤或短少。
處理	執行計算或資料的整理。
輸出	執行該程式處理結果的輸出工作。
錯誤掌理	執行錯誤狀況的分析並輸出錯誤訊息。在小程式裏錯誤的掌理可以併到其他模組內。
關閉程序	執行該程式執行的結束工作。

在一個程式裡，這些模組形成一個邏輯層次圖（如圖1-2）。一個程式以主線控制模組開始，處理的順序是順著輸入、輸入檢核、各個處理模組及輸出（這些都在主線控制模組控制之下）的邏輯順序進行。錯誤掌理若在執行上述任何一個模組發現有錯誤發生時所需要的模組，也可以併入各該模組之內。在處理結束時，就執行關閉程序以結束該程式。上述各種功能在良好撰寫的程式裡通常都存在著，但是並不一定寫成個別的模組，爲了適應各該程式的流程，他們可能合併或作各種不同的組合。

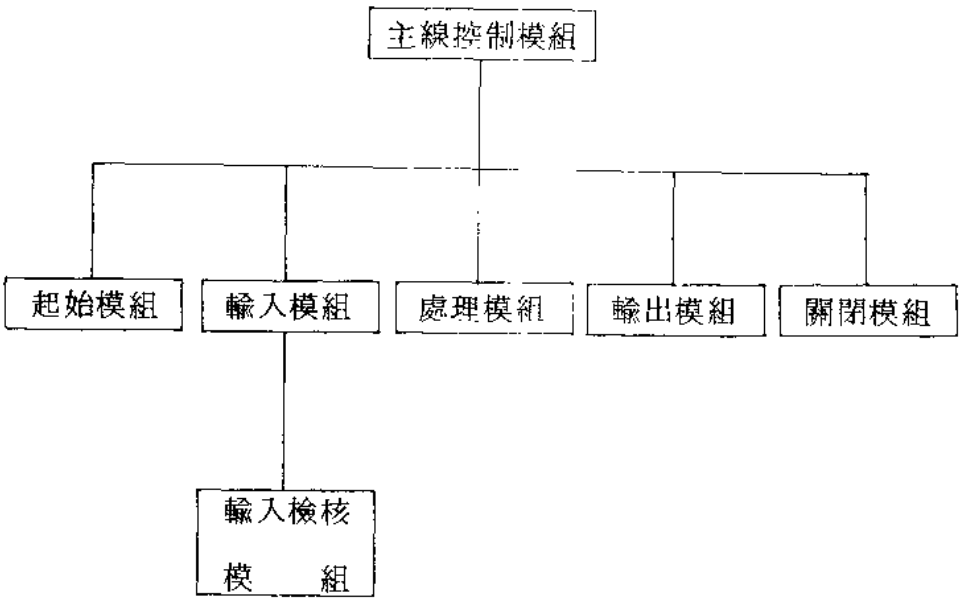


圖 2-2　一個程式中的邏輯層次模組

1-3 結構化程式設計

爲了達到正確、抵擋錯誤、可維護程式的目標，有一種方法——**結構化程式設計**——是用直截了當、容易瞭解的方式來撰寫程式。

1-3-1 基本撰寫結構

任何電腦程式都能夠僅用三種邏輯結構或是這三種結構的組合來撰寫：

1. **循序結構**。
2. **選擇結構**。
3. **重複結構**。

這三種結構在規則化的程式設計上很有用，因爲：

(1) **程式簡化**：每一種結構、以及由這三種結構所組成的模組都是只有一個入口和一個出口，因此程式的複雜性就簡化了。

(2) **容易維護**：利用這三種結構所撰寫的程式，可以由上而下閱讀下來，使得程式的邏輯更明白的顯現出來，因此在除錯和維護時要容易得多。

1. **循序結構**：

循序結構是照順序執行兩種功能，也即是一個功能接一

個功能。如圖1-3；控制流程是執行功能A之後循序執行功能B。

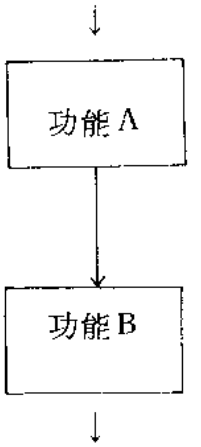


圖1-3　循序結構

2. **選擇結構**：

選擇結構是由一個條件和兩個相異路徑的功能所組成，如圖1-4，當條件P成立時執行功能C，否則執行功能D，在執行完畢任何一個功能之後走向共同的出口。此種結構又叫做“若……則……否則……”（IF……THEN……ELSE……）結構。

3. **重複結構**：

重複結構也可以叫做循環（Loop），當條件不成立時，循環裡的功能E就重複執行一直到條件成立爲止。

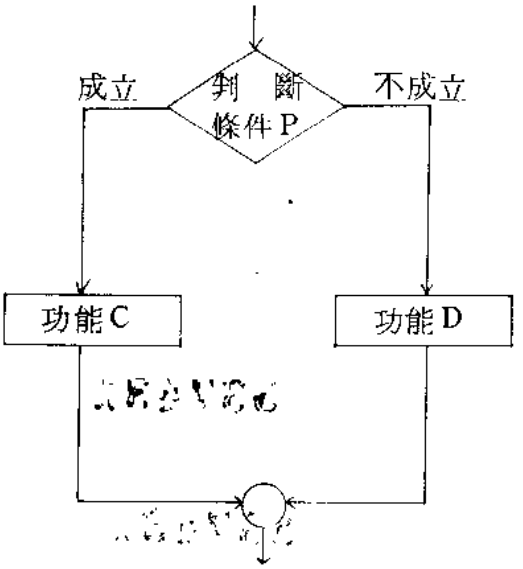


圖 1-4　選擇結構

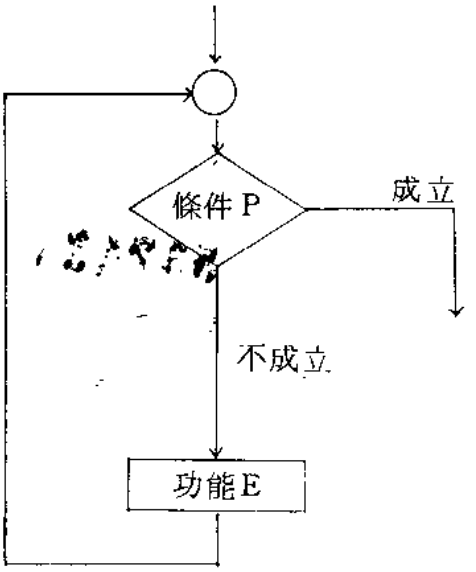


圖 1-5　PERFORM UNTIL 重複結構