商用程式語言 COBOL 第13版

鍾英明・陳秋發 合著　吳建平 校閱

博文出版社出版

商用程式語言
COBOL

鍾英明 陳秋發 著

吳建平 校閱

博文出版社出版

商用程式語言
COBOL

編　　者：鍾英明　陳秋發
出 版 者：博　文　出　版　社
發 行 者：博　文　出　版　社
九龍大連排道872號
印 刷 者：鴻　文　印　刷　廠
香港柴灣工廠大廈五樓

定價：H. K. $

序

這是一本爲初學者撰寫的計算機商用語言書籍，建平能爲它作校訂工作，深感愉快。

作者鍾英明先生及陳秋發先生皆爲計算機科學專家，都具備十多年之教學及實際工作經驗，以流利的文筆，淺出的手法，編著此書。此書結構適合初學者，每一子題皆有詳細之說明及簡短之範例。在適當之處，對前述之子題再予以交叉應用，因此無零碎之缺點，而有前後相輔相成之優點，使學習者隨着進度愈得以體會其要領。

本書適合於大學及專科學校學生作教科書及自習之用，讀者可無需任何計算機語言之基礎。本書取材精闢內容充實，可引導初學者建立設計有效之COBOL 程式之能力，瞭解如何應用 COBOL 於資料處理，以及具備設計複雜程式之邏輯思考能力。

吳建平　謹識

序

COBOL 程式語言爲當今計算機程式語言中，使用最爲廣泛者之一，雖然它亦屬於高級程式語言，但因其輸出入作業功能之要求繁多，對一般讀者來說，要比研修 FORTRAN 費心一些，因此常有讀者表示使用 COBOL 原文教材，難能完全理解其中意義，致使只要好好學習一個月便能熟巧之程式語言，竟因文字上的障礙，而耗時費力尚不易通達，此乃筆者編寫本書之動機和目的。

本書之語法以 IBM 最新之 370 機器系統爲主，在每一章節之後，除有一般之習題外，並列有附帶答案的練習，而且事例簡明，前後完整，相信必能讓讀者在短時間內獲致通達。

著者　識

目　錄

第四章　設備部

第五章　資料部

第六章　程序部

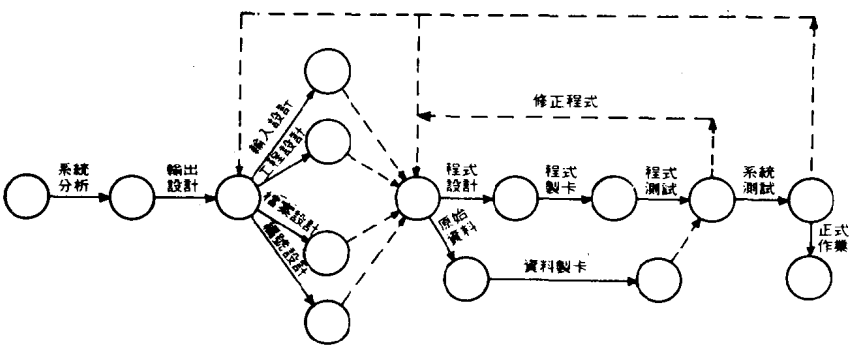
第一章　電子資料處理系統之基本概念

最近十年來，由於科學技術的突飛猛進，已促使電子計算機的發展步入了另一新的境界，它的性能不斷的在增强，而其價格成本却有日愈下降之趨勢，這便是科技進步的正常結果。

目前電子計算機之使用範圍和利用對象已經廣泛到無可限界的程度，不但普遍的用於工商業界，今後人類大部份的活動形態也都將因其而受到改變。爲了迎接此一新的局面，確實值得吾人對其作一研究和探討。

1-1 EDPS之作業程序

電子計算機的功能雖大，但它必竟是一工具，用得委當獲利無窮，用之不當不但浪費金錢，同時將帶來麻煩。一件事情或一種作業在使用計算機時，必須依照某一程序，在此稱爲電子資料處理系統（Electronic Data Processing System）之作業程序，其情形如下：



(1)系統分析：系統分析主要是爲了解所要進行的作業。包括爲何以機器來代替人工作業，此作業必須產生何種結果？以及作業過程中將受到的各種限制等。唯有了解作業系統之後，才能針對問題設計各種要素。

(2)輸出設計：輸出爲作業的目標，目標未定之前其他的設計無從定論。輸出設計包括決定產生輸出的周期爲年、月或日；輸出的方式爲報表或其他的媒體；輸出的資料項目及項目的位數，字形等等。

目標確定之後，開始其他產生此一目標的要件設計。

(3)輸入設計：欲達到輸出的目標，最少應使用何種原始資料？若將輸出視爲成品，則輸入有如製造此成品的原料。

(4)工程設計：設計作業的流程，各處理步驟應如何的配合才能達到作業目標，節省作業成本和提高作業時效。

(5)檔案設計：包括各輸入、輸出檔的格式，檔中每一錄的項目及其位置等，如人事主檔、庫存主檔等之設計。

(6)編號設計：對某些不適於直接以其原始形態來處理的資料，如員工的姓名，貨品名稱等等，如何給予一適當的編號，使處理過程更爲簡便。例如，由學生的學號便可知其入學年份、系級等；由員工編號亦可知其所屬部門、職位等。

有了以上的設計，開始步入處理細節的設計

(7)程式設計：程式是一套指揮計算機如何去處理資料的指令。各作業步驟的程式不但要達成卽定的結果，也要有效的利用機器的特性，減少作業的時間。

(8)程式製卡：將寫好的程式製成卡片，作爲計算機接受程式指令的輸入媒體。

(9)程式測試：利用能產生各種情況的假設資料測試程式，以保證程式所處理的結果正確無誤。

程式只是個別作業步驟的資料處理而已，程式的正確並不能保證各

作業步驟間能夠互相的配合，而且，某些遺漏的處理步驟並不一定能在程式測試時發現。因此，在正式開始作業之前，必須以部份的眞實資料來對系統的整體作一測試。

⑽原始資料：準備完整的原始資料，一方面以其中一部份作系統測試；另一方面，當系統測試無誤後便可開始正式作業。通常，爲了爭取時效，(2)～(6)的設計確定無誤後，原始資料的準備及程式設計便可開始平行作業。

⑾資料製卡：將原始資料儲存於計算機可接受的媒體上。這包括資料的製卡或直接輸入於磁帶上。

⑿系統測試：在正式作業前對系統作一整體性的測試以確定系統之功能是否符合理想。

綜觀上述各作業步驟，最花費人力時間的可能是原始資料的收集及資料製卡階段，因此，對原始資料內容的確定（輸入設計）必須特別的愼重。同時，設計時必須考慮到將來系統的擴張，預先準備充分的原始資料項目。

1-2 程式語言的種類

程式語言可分爲三大類：機器語言、組合語言與高級語言。

1.2.1 機器語言

機器語言（Machine Language）是中央處理機內部使用的語言，它的指令是由一連串具有意義的 0 與 1 所組成。由於這種語言不易了解，程式書寫不易，只有在40年代電子計算機出現的初期使用過，隨後即被其他語言所取代。

現在，我們雖不直接以機器語言來書寫程式，但它仍是中央處理機唯一能辨識的語言，以其他語言所書寫的程式必須經過**編譯程式**（Compiler）的編譯，轉換成機器語言後，中央處理機才能執行。

1.2.2 組合語言

組合語言（Assembly Language）與機器語言非常接近，它是以

一些程式設計人員容易記憶辨認的代號來代替機器語言的0與1。例如，相加指令的機器語言爲01011010，而組合語言却以A來表示。在程式執行之前，組合語言必須以其**組合程式**（Assembler）將之轉換爲機器語言，才爲中央處理機所接受。

組合語言雖較機器語言更爲方便，其程式仍稍嫌繁雜，一般的應用程式較少使用。但因其能控制中央處理機的每一細部動作，在控制機器作業的系統程式及一些講求效率的常用程式中，常使用組合語言書寫。

1.2.3 高級語言

高級語言（High Level Language）是專爲各種應用而設計的語言，目前已有一百餘種。除了具有特殊用途的語言外，現被廣泛使用的通用語言有FORTRAN，ALGOL，COBOL，PL／1，RPG等。

高級語言必須經過其編譯程式的轉換，成爲機器語言後才能執行。它與組合語言不太相同。組合語言與機器語言的關係差不多是一對一的，也就是，每一個組合語言的指令對應的產生一個機器語言的指令。但高級語言却是爲方便使用者而設計，一個簡單的敍述可能要由數十個，甚至數百個，數千個的機器語言指令來完成。正因爲如此，以高級語言寫程式時，設計人員更能專注於處理的邏輯，減少對機器內部許多細節的考慮。

高級語言的敍述以類似常用的英語來表達，簡單易學，使許多對機器內部的複雜情況並不了解的使用者亦能應用自如。以下將介紹目前較通用的幾種語言。

FORTRAN（FORmula TRANslating language）

FORTRAN是IBM公司於1956年，針對科學及工程計算而設計的語言。顧名思義，其敍述類似數學的式子，易懂易學。由於FORTRAN能簡易的表達計算的過程，目前大部份涉及計算問題的程式都以之書寫，是流傳最廣的一種語言。

ALGOL（ALGOrithmic Language ）

ALGOL 是一種口語化的程式語言，其功能與FORTRAN相似，於1958年由德國的GAMM計算機公司及美國計算機協會（ＡＣＭ）所共同發表。當初設計ALGOL的原意是希望以此來統一計算機的程式語言，後來雖然失敗了，設計ALGOL的經驗却奠定了以後發展程式語言的基礎。目前，ALGOL語言在歐洲一帶流行較廣。

COBOL（ COmmon Business Oriented Language ）

COBOL是爲商業資料處理而設計的語言，由美國的資料系統語言審議會（ CODASYL）所編定，於1960年發表。有關COBOL的特徵及形式請參閱本章1－3節。

PL／I（ Programming Language／1 ）

FORTRAN主要用於處理數值資料，而COBOL的主要對象却是非數值資料，PL／I是綜合兩者之優點而設計的一種語言，於1964年由IBM公司及其一大用戶SHARE公司所共同發展的。由於功能較强，用途廣泛，其結構也較其他的程式語言複雜，比較難學。目前，同時涉及複雜的數值及非數值處理的作業並不多，因此PL／I尚少爲人們所用。

RPG（Report Program Generator）

RPG是專爲產生報表而設計的語言。與其他的語言相比，其性質較爲特殊。這種語言並沒有類似數學式子或英語的敍述，程式的書寫是採取填表式的，設計人員只須在固定的表格填上適當的符號，便可表達其資料的處理流程。RPG的功能相當强，目前在商業資料處理方面流傳甚廣。其缺點是對這種語言不熟悉的人不容易了解其程式內容。

一般而言，高級語言都是針對某一方面的應用而設計的，其程式也較專注於問題的處理流程，盡量減少其他不必要的考慮。因此，對計算

機沒有深入了解的人亦能輕易的使用。此外，高級語言具有易懂易學的特點，通常在一星期以內便可掌握其概念，再對各式各樣的問題，由淺入深的約寫十個練習程式，便可應用自如。當然，想非常熟練的使用此語言還有待更進一步磨練，從經驗中吸取各種技巧，才能使所設計的程式日趨完美。

1-3 COBOL程式之特徵及形式

本節將介紹COBOL 程式的特徵及其形式，使讀者在開始學習之前對COBOL 有一整體的概念。

1.3.1 COBOL 程式的特徵

COBOL 是針對商業資料的處理而設計的。商業資料的特點是數量龐大，且大部份屬於非數值性，數值方面的處理却較爲簡單，通常只用到算術的四則運算而已。因此，處理商業資料的語言需有下列特點：

1. 高效率的輸出入作業，俾能應付龐大的資料量。
2. 能簡易的控制主儲存體之位址，强化處理非數值資料的性能。
3. 簡單易學，即使沒有數理基礎的人也能應用。

此外，由於經濟的快速發展，商業資料之處理過程亦經常變更，商用語言必須具備適應發展的能力。即

4. 程式的結構必須段落分明，容易維護，更改程式的某一部份不會使其整體受到太大的影響。
5. 語言的敍述須口語化，可當作參考文件用。

COBOL 就是因應上述各種要求而設計的語言，自頒行後廣受歡迎，目前大部份商業資料處理的程式都使用COBOL 語言。

1.3.2 COBOL 程式的形式

COBOL 程式中以類似英語的敍述，表達資料在主儲存體中的狀況，以及處理資料的過程。程式可分爲四大部份：識別部、設備部、資料部及程序部。每一部又可分爲數節，節中再分段。下面爲一完整的COBOL 程式：

```
SEQUENCE                A  B  COBOL STATEMENT
ΦΦ1 01  IDENTIFICATION DIVISION.
ΦΦ1 02  PROGRAM-ID. OVER-FEE-LISTING.
    03
    04  ENVIRONMENT DIVISION.
    05  CONFIGURATION SECTION.
    06  SOURCE-COMPUTER. IBM-37Φ.
    07  OBJECT-COMPUTER. IBM-37Φ.
    08  INPUT-OUTPUT SECTION.
    09  FILE-CONTROL.
    10      SELECT OVER-HOUR-FILE
    11          ASSIGN TO SYSΦΦ9-UR-35Φ5-S.
    12      SELECT OVER-FEE-FILE
    13          ASSIGN TO SYSΦ1Ø-UT-341Φ-S.
    14
    15  DATA DIVISION.
    16  FILE SECTION.
    17  FD  OVER-HOUR-FILE
    18      LABEL RECORDS ARE OMITTED.
    19  Φ1  OVER-HOUR-RECORD.
    20      Φ2  IN-EMPLOYEE-NO      PICTURE IS 999999.
    21      Φ2  IN-OVER-HOURS       PICTURE IS 99.

ΦΦ2 01  FD  OVER-FEE-FILE
ΦΦ2 02      LABEL RECORDS ARE STANDARD.
    03  Φ1  OVER-FEE-RECORD.
    04      Φ2  OUT-EMPLOYEE-NO     PICTURE IS 999999.
    05      Φ2  OUT-OVER-HOURS      PICTURE IS 99.
    06      Φ2  OUT-OVER-FEE        PICTURE IS 9999.
    07
    08  PROCEDURE DIVISION.
    09  BEGIN.
    10      OPEN INPUT OVER-HOUR-FILE
    11           OUTPUT OVER-FEE-FILE.
    12  FEE-CALCULATION.
    13      READ OVER-HOUR-FILE
    14          AT END GO TO END-OF-JOB.
    15      MOVE IN-EMPLOYEE-NO TO OUT-EMPLOYEE-NO.
    16      MOVE IN-OVER-HOURS  TO OUT-OVER-HOURS.
    17      IF IN-OVER-HOURS GREATER THAN 3Φ
    18      THEN COMPUTE OUT-OVER-FEE
    19          = 15Φ + 1ΦΦ * (IN-OVER-HOURS - 3Φ)
    20      ELSE COMPUTE OUT-OVER-FEE
    21          = 5Φ * IN-OVER-HOURS.
    22      WRITE OVER-FEE-RECORD.
    23      GO TO FEE-CALCULATION.

ΦΦ3 01  END-OF-JOB.
ΦΦ3 02      CLOSE OVER-HOUR-FILE,
    03            OVER-FEE-FILE.
    04      STOP RUN.
```