程式設計講義

第一回

60772A-1

社團法人 考友社 出版發行

程式設計講義　第一回

目錄

第一講　程式設計概念

命題大綱

一、程式設計的意義
- ㈠程式語言的意義
- ㈡程式語言的設計
- ㈢程式設計之步驟
- ㈣程式語言之設計及評估標準比較
- ㈤程式設計之目的

二、程式語言的分類
- ㈠第一代程式語言－機器語言
- ㈡第二代程式語言－組合語言
- ㈢第三代程式語言－高階語言
- ㈣第四代程式語言－超高階語言
- ㈤第五代程式語言－自然語言

三、高階程式語言
- ㈠高階語言之處理器
- ㈡高階語言之優點
- ㈢高階語言之類型
- ㈣高階語言之實作

四、各種語言的一般語法
- ㈠語法元素
- ㈡語法分類

五、BNF 語法
- ㈠符號
- ㈡產生規則的格式與合併
- ㈢推導過程
- ㈣剖析樹

六、模稜兩可文法
- ㈠定義

重點整理

一、程式語言的意義

㈠程式語言的意義：

1.程式語言是由一組系統化的符號所組成之集合，可指揮電腦執行指定的動作。

2.程式（program）＝演算法（algorithm）＋資料結構（data structure），凡是能描述演算法和資料結構的系統化符號即稱爲「程式語言（programming language）」。

3.學習程式語言的意義：

⑴對所使用的程式語言更加瞭解。

⑵改進程式的架構。

⑶可選擇合適的程式語言。

⑷能更容易學習新的程式語言。

⑸較容易設計出新的程式語言。

㈡程式語言的設計：

1.程式語言設計的目標係爲達到簡潔（simplicity）之要求。

2.程式語言設計之標準：

表(一) 程式語言設計之標準

可讀性	定意良好的語法和語意
可靠性	可靠性高表示語法不易發生邏輯錯誤，若有錯誤也易發現
獨立性	語言的數種特性間應彼此獨立，不相互干擾
可證性	若語言特性有正式定義，加上純熟的數學基礎，即可驗證程式
一般性	即所有特性皆可由一組簡單的概念組成
快速翻譯	設計程式語言時若能考慮語法和語意，較易建立處理次數較少的編譯器，達到快速翻譯之目的
高效率的目的碼	編譯器和對應語言的複雜程度並非正比關係，而是更加複雜，維持語法的簡單性可產生高效率的目的碼

通用記號的一致性	同一個運算元可用在多種型別的運算表示中
子集	瞭解語言的部分即可運用
齊一性	程式碼相似者具有相似意義
擴充性	具有建立新的抽象化資料型別和運算符號之能力

㈢程式設計之步驟

1.描述問題：

定義要解決的問題，確定其範圍和限制。

2.分析問題：

⑴針對要解決的問題進行分析，將問題分解成許多較小的問題，較小的問題再依序分解爲更小的問題。

⑵分析各問題的特性、類型及彼此的關聯，做爲解題的依據。

3.推導及設計解題方案：

⑴依分析結果將解題的過程分解爲一系列有次序的步驟，並敘述每一步驟的處理邏輯。

⑵上述「一系列有次序的解題步驟」稱爲「解題方案」或「演算法（algorithm）」。

4.細部邏輯設計：

利用適當的程式設計工具（如虛擬碼、流程圖等），將「演算法」中每一個步驟，依其先後次序詳細、明確得敘述。

5.程式的撰寫、編譯、連結及執行：

⑴編寫原始程式（coding）：選擇適當的程式語言，參照程式的細部邏輯，撰寫「原始程式（source program）」。

⑵輸入或讀入原始程式（key in　or input）：

將原始程式輸入記憶體，或將已儲存在磁碟（或磁帶）的原始程式讀入記憶體，準備編譯。

⑶編譯原始程式（compilation）：

將原始程式編譯爲「目的程式（object program）」，此時若發現程式有錯誤，即修改原始程式，再重新編譯，至完全正確爲止。

⑷連結目的程式（linking）：

因編譯過程中並未處理原始程式所引用到的輸入常式（I/O routine）、函數（function）、副程式（subroutine）之位址，故產生之目的程式仍無法執行，須經由連結程式（linker）將上述各項

的位址連結至目的程式，以得到可執行之目的程式。

⑸載入（loading）與執行（execution）：

利用載入程式（loader）將可執行之目的程式由輔助儲存體（如磁碟等）載入主記憶體中。

6.程式的測試及除錯：

⑴可執行的程式需輸入測試資料進行「測試（testing）」，檢查程式中是否存在「邏輯錯誤（logical error）」，若有則須進行「除錯（debugging）」，找出錯誤發生處、加以改正。

⑵程式除錯的方式：

①檢查是否有硬體設備不穩定、執行環境不同等外在因素，造成程式執行發生錯誤。

②若無上述情形，則檢查程式中是否有以下錯誤：

A.鍵錯程式或資料。

B.未設定變數起始值。

C.發生「溢位（overflow）」。

【註】如某數除以零。

D.未處理異常或例外情況。

E.程式的處理邏輯錯誤。

F.資料的格式不符。

G.誤用同一程式的其他版本。

③幫助除錯：

A.從程式編譯時產生的「錯誤列表（listing）」中找出程式指令語法上的錯誤，並修正。

B.利用系統提供的「追蹤（trace）」功能協助除錯。

【註】追蹤的功能可顯示有關變數執行前及執行後之值、程式執行路徑等。

C.「傾印（dump）」出記憶體中某一段範圍或檔案內容，供偵錯參考。

7.編製程式說明文件：

⑴程式說明文件（document）對於程式之維護及發展非常重要。

⑵詳細的程式說明文件包括：

①在程式中加入適當的註解（comment），提高程式的可讀性。

精選試題

一、試說明使高階語言的程式能在計算機上執行，須經過的步驟？

答：㈠編寫原始程式：

選擇適當的程式語言，參照程式的細部邏輯，撰寫原始程式。

㈡輸入或讀入原始程式：

將原始程式由鍵盤輸入記憶體，或將已儲存在磁碟的原始程式讀入記憶體。

㈢編譯原始程式：

將原始程式編譯爲「目的程式（object program）」，此時若發現程式有錯誤，即修改原始程式，再重新編譯，至完全正確爲止。

㈣連結目的程式：

因編譯過程中並未處理原始程式所引用到的輸入常式（I/O routine）、函數（function）、副程式（subroutine）之位址，故產生之目的程式仍無法執行，須經由連結程式（linker）將上述各項的位址連結至目的程式，以得到可執行之目的程式。

㈤載入與執行：

利用載入程式（loader）將可執行之目的程式由輔助儲存體（如磁碟等）載入主記憶體中。

㈥程式的測試及除錯：

可執行的程式需輸入測試資料進行「測試（testing）」，檢查程式中是否存在「邏輯錯誤（logical error）」，若有則須進行「除錯（debugging）」，找出錯誤發生處、加以改正。

二、試說明「組合語言」與「高階語言」的基本差異。

答：組合語言爲一種與機器相關（machine dependent）的語言，移植能力（portable）差，一般僅適用於某些特定廠牌、機型的電腦，但效率較佳。

高階語言爲一種與機器無關（machine independent）的語言，具移植性，可適用於各種機型，彈性較大，但執行效率較差。

三、參考下列程式句型，寫出其相對應的語意（operational semantics）。

```
switch(expr){
```