數學與程式設計

吳維漢

國立中央大學數學系

2012-09-07

常有人問：寫程式須不須先要將數學學好？或者問：數學學不好，是否程式就寫不好？可見許多人即使不喜歡數學，為了學好程式，仍對數學念念不忘。在我看來，**寫好程式的最基本要素就是「能靈活運用思考邏輯的能力」**。這裡的「思考邏輯」剛好是數學人所最重視的素養，也是在學習數學過程中一直在培養訓練的能力。「數學」學得好，邏輯觀念一定清楚，能靈活運用邏輯與數學來推論問題的前後次序或關聯，自然具備學好程式設計的基本要素。但有些人數學並沒有學得很好，卻仍將程式寫得很好，這些人應本身已養成隨時運用思考邏輯的習慣來解決問題，因而不需再經過數學的訓練。事實上，若能靈活地運用思考邏輯，是可以同時寫好程式與學好數學。思考邏輯能力弱的人，在經過數學的嚴緊訓練，是可以提昇的。

我常觀察到有些人的數學似乎學得不錯，但一直苦於無法將程式學好。究其原因，這些人的思考邏輯能力可能無意中被自己「關在」數學學科內，沒有將它應用到程式設計上，這種現象多半與其從小所接受的教育方式有關。在臺灣多數的學生從國中以後，所學的數學都是用來考試，不是用在解決與日常生活有關的數學問題，連帶使得許多人只將「思考邏輯」應用於學科考試中。特別是有些國高中的數學題目極盡刁鑽古怪之能事，使得學生認為學習數學所培養的「思考邏輯」的唯一用處就是用來解決這些數學問題。由於每次運用思考邏輯的過程都極耗心力，因此離開了考試，思考邏輯就不自主地被封印起來，休養等待下次考試的來臨。一個被封印思考邏輯的腦袋是無法用來推敲極度講究前後邏輯順序的程式問題，如此自然無法學好程式設計。

解決的方式首先要由心態著手，要解放被自己不自覺封印起來的思考邏輯，將其應用在程式設計中。在作法上可先試著將每一道程式題目當成數學題目來練習，但與傳統考試不同的是：在傳統考試中，解決數學題目所需的條件都已在試卷上寫得清清楚楚，不需自行尋找。但在撰寫程式時，所有的條件都被隱藏起來，需要自行尋找。此時就只能運用邏輯思維，一步步仔細地由答案逆向推演，找出步驟之間的關係與對應條件，試著將思考邏輯解放出來當成你在程式設計的朋友。

Old habits die hard，要立即解放已被塵封許久的思考邏輯並馬上應用於程式設計中，大概也須經過一段時間的練習。在轉變的過程可能充滿挫折，但只要持之以恆，自然能突破難關，掌握到其中要訣。此外初學者切忌以三天打漁，兩天曬網的方式學習程式設計，這種斷斷續續的學習方式經常會將已快要領悟突破的腦袋又打回原樣，徒然浪費更多學習時間。當你能自由地運用思考邏輯時，之後每當在程式設計中遇到困難時，你的「朋友」就會主動替你解決問題，找尋解決方式，此時程式設計就不再是件痛苦的事，取代的是完成程式後的喜悅。

數學在程式設計的過程中如影隨形，舉例來說，若要寫個程式列印以下高度為 n 的山形（假設在最下列的星號之前無空格）：

```
   *
  ***
 *****
*******
```

如果你完全沒有頭緒就代表你忘了數學的存在。首先要注意的是程式列印的順序是由上而下，由左向右。程式須對每個輸出字元完全負責，包含不顯示任何東西的空白字元。若將題目當成數學題目，以縱行方向為 i，橫列方向為 j，則每一橫列依次列印 x 個空格， y 個星號 ('*')，最後再加上一個換行字元 ('\n') 讓游標跳到下一列的起始位置。x 若以函數表示可寫成 x(i,j)，y 可寫成 y(i,j)。當 i 由 1 遞增到 n，觀察圖形，x(i,j) = n-i，y(i,j) = 2i-1。若改用文字表達執行步驟可寫成：

```
讓 i 由 1 到 n 遞增，每次執行以下步驟：
  (1) 印出 n-i 個空格
  (2) 印出 2i-1 個星號
  (3) 印出一個換行字元
```

轉成對應的程式碼[1]可寫成：

```
for ( i = 1 ; i <= n ; ++i ) {
    for ( k = 1 ; k <= n-i ; ++k ) cout << ' ' ;
    for ( k = 1 ; k <= 2*i-1 ; ++k ) cout << '*' ;
    cout << '\n' ;
}
```

當你學會了如何以數學協助你列印一座山的問題後，你可以試著將之擴展為列印 n 座相連的山，或是在底部加上倒立的山成為一顆鑽石，然後再將之擴展為列印出一排鑽石，或 n 排鑽石。每次完成一個題目，就再加些新的條件到問題中，然後想想要如何

[1] for 為迴圈式子，可重複執行，++i 為 i 遞增一，cout << 可印出其後的內容

更動你先前的數學式子與改寫程式碼。一步步訓練你的數學邏輯思維，讓它主動出面替你解決問題，持續一段時間後，數學式的思考自然就會與程式設計融合在一起。

初學程式設計的人最好要養成使用紙筆推導公式的習慣，不要讓你的「朋友」都隱身在腦海中以「運籌帷幄」的方式幫你寫程式，這很容易出錯。對初級的問題而言，推導公式通常僅需國中程度的數學，耗費數分鐘的時間。若因此而偷懶省略，經常要付出更多的時間代價，進而打擊學習程式設計的意志。事實上，初學者也可透過在紙張的推導順勢整理思緒，檢驗步驟的前後順序邏輯是否正確。據我的觀察，許多半途而廢的人，通常是學習程式設計的方法出了問題。聰明的程式設計師是會先用紙筆導公式，描繪出大致的步驟流程，之後才在螢幕前撰寫程式。學習程式的過程一定要 smart，不要盡是 work hard，因為後者通常撐不了多久。

擅長撰寫程式的人，若不懂得一些專業數學理論是無法進一步開發某些特殊的軟體程式。例如：三維電玩遊戲與一些繪圖程式經常須要變動點、線、面的位置，這些都需要利用到矩陣運算。一些科學應用，例如：氣象報告時常看見的等壓線或等雨量線，需要使用數值分析的插分法。若沒有數論的密碼學，網路商務就不會存在。列印文件時所看見的各種漂亮的描邊字型沒有使用數學方程式也不行。車用的衛星導航系統須要特殊的演算法才能規畫出適當的行車路線。又如在網際網路傳輸檔案時，為減少傳遞時間，通常會先將檔案壓縮後傳輸，到達目的後才解壓還原，這個壓縮與還原檔案的方法也需要數學。以上這些都是很常見的數學應用，沒有學好某些領域的數學，光會寫程式是不夠的。在整個資訊科技發展的過程中，數學一直都是隱身在其後的有力推手。

並不是所有數學人都懂得前述應用背後所隱藏的各種數學理論，但對經過長時間接受數學專業訓練的學生而言，要讀懂數學理論一定比非數學專長的人容易得多，這是數學系學生的最大優勢。在我看來，數學如同一把寶劍，唸數學的過程有如在磨手中的寶劍，數學學得好，劍刃越是銳利。學習程式設計就是教你如何舞動這把劍，讓它發揮作用，上場對陣時，可以招招克敵致勝。一個懂得程式設計的數學人在職場是極具有競爭優勢，千萬不要小看「數學」與「程式設計」合體後的職場致勝組合。