# 程式自學十年心得：想吃這行飯，學好演算法與資料結構才能讓你站穩腳步 | TechOrange 科技報橘

buzzorange.com/techorange/2016/04/13/self-study-program-with-datastructure-and-algorithm

TO 編輯推薦觀點 2016年4月13日

> 《TO 導讀》：自學程式設計已成為風潮，不僅坊間教材眾多，只要連上網路，還有無限多的學習資源能接觸。但是資源豐沛代表的也是龐雜混沌，選擇自學的初學者也容易因此迷失方向。事倍功半還算事小，更可能就這樣失去了自學的動力。本文作者雖然是文組學生，但自學程式已有十年經歷，工作也與程式相關。他根據自身經驗分享自學心得，以下由作者第一人稱描述。

前言

從國中開始自學程式到現在，已經有十年了，所以我應該算是個資深的自學者吧 XD。有滿多心得想跟大家分享，這篇主要是分享一些我認為初學者可以參考的學習方向

我的背景是從國中自學程式，大學念文科，現在出社會靠寫程式維生，工作經歷一年

正文

首先，沒有最好的程式語言，只有最適合的程式語言。程式語言本來就只是工具、只是手段，從來都不是重點，重點是：你的目的是什麼？你要解決的問題是什麼？

假設你今天想要寫一個網頁留言板（這邊只討論後端，不包含前端），那你就不應該用 C 來寫，因為比起 C，php 會更適合。

你可能會問：那 nodejs, rails, asp.net… 就不適合嗎？如果你只是「單純想寫個網頁留言板」，其實上面這些都很適合，挑一個你最喜歡的就好。但如果你要解決的問題不一樣，就應該重新思考。

例如說，「我想要超級快速的寫個網頁留言板」，你就應該用 rails，因為 rails 開發真的超級快，隨便打幾個 command，你的網頁就神奇的出現了！「我想要很潮的網頁留言板！」，那你可以考慮 Go, Swift, Nodejs 等等

換個例子，若是你今天想要「研究電腦較底層的運作」，你應該要學 C，或是學組合語言。因為比起其他高階語言，這兩種絕對會讓你更了解底層到底在做什麼。

其實以上這些只是想提醒你：不要為了學程式而學程式，程式只是手段、手段！重點在於目的！重點在於你想要透過程式，達成什麼樣的目標（如果你真的只是純粹喜歡寫程式，當然也是可以啦 XD）

所以，要想清楚你為什麼要學程式？

我的理由很簡單，我覺得寫程式讓我很有成就感，只要幾天就可以寫出一個網頁。可以幫自己寫部落格系統、幫系上寫投票網站之類的，儘管寫出來的東西可能不怎麼樣，但從無到有，是很感人的一件事。我永遠忘不了我用 VB 拉幾個按鈕出來，點下去就可以跑出文字的那瞬間，我有多感動。那是我第一支程式。

無論是單純有興趣或是只想要轉職，以下的幾項建議是我覺得能幫助到你的，這幾點我「基本上」是按照順序列的，也就是一個完成之後再完成下一項。但也有些例外，是可以隨時都學的

學英文

這個已經在版上講到爛掉。Q: 我該學什麼程式語言 A: 英文

為什麼？因為英文真的很重要！聽說讀寫，至少讀要讀的懂，你才看的懂那些外國公司的 API 文件，發生問題的時候，才能去 stackoverflow 上面發問。講到提問，推薦這篇：〈提問的藝術〉，大家都會問問題，但有些問題不會有人想回答，像是隔空抓藥（「我寫了一個網路通訊的程式，想要讓 A 手機跟 B 手機可以傳訊息，可是按了卻沒反應，怎麼辦？」）

或是排版爛的程式碼

```
String a="hello";
String bb= "world" ;
for(int i=0;i<100;i++){
if(socket.alive){
socket.send(a);
}
}
```

一個好的問題，或是說，讓人看的懂得問題，應該要包含四點：

> 1. 目的（你的程式要達成什麼目的？）
> 2. 手段（你怎麼達成你的目的？）
> 3. 錯誤（在什麼場景下發生的什麼錯誤？）
> 4. 程式碼（有排版過的程式碼，拜託）

以上問題可以修正成：「我想要讓 A 手機跟 B 手機傳遞訊息，我用的是 java 的 socket，可是按下發送時出現：java.net.SocketException: Connection refused: connect，想請問一下有可能是錯在哪裡？部分程式碼：https://gist.github.com/xxxxxx」上面的問題還可以再修正、補充更多細節，但至少比剛開始那個好滿多的，會讓人比較想回答。

程式碼一定要找適合的地方貼，不知道貼哪裡的話，google：貼程式碼，就有一堆文章讓你參考

好，我們繼續回到學英文這點 XD，英文不好的話，中文的教學資源其實也很多，尤其是簡體中文，有很多高質量或是新手教學文都很不錯。但畢竟較新的技術，或是較多人氣的地方用的還是英文英文好的話絕對不吃虧，可以運用更多資源。

學會 google

別懷疑，這真的是超重要的能力，我時常覺得，會 google 就先贏一半了。現在網路資源這麼多，有時候發生你看不懂的 Error，完全沒頭緒，你就直接丟 google，就批哩啪啦一堆解法出現了。這時候除了很開心的 copy paste 以外，有一點很重要：「試著理解它」。若是不理解，你就只是個只會複製貼上的機器人，一點成長都沒有，至少要理解到兩點：為什麼有這問題、這個解法是如何解決的

關鍵字的使用也很重要。我在搜尋的時候很常利用空白分割，不讓語句太死，例如說你想找：程式入門教學，我會搜尋：程式 入門。想要找：在網頁上如何實作拖曳效果，我會搜尋：網頁 拖曳。

google 的能力可以慢慢培養，搜尋多了你就會有那種感覺了

學 scratch

我聽過這個很多次，但實際接觸才知道「真的很好用」。這個就是讓你用很簡單、很視覺化的方式來寫程式，你只要拖拉幾個拼圖、組合在一起，就可以寫出一隻可以動的程式，甚至是一個小遊戲！

隨堂測驗：你要怎麼開始學 scratch？

如果是我的話，就會搜尋：scratch 教學，或是 scratch 入門，這邊直接幫大家附上兩篇排名較前面的結果〈scratch 程式教學〉、〈Scratch 教學的第 1 堂課（1/5）適合親子共學的兒童程式設計入門〉。

為什麼要學 scratch 呢？前面有說到，每種程式語言都有不同的適合的場景，還有一點是，「每種程式語言，都有很大部分是相同的」，例如說：變數、函式、迴圈、條件判斷、陣列…

你學 C 有這些、Javascript 有這些、Swift 有這些。不管學哪種程式，都會碰到這些最基本的東西，無論是再複雜的程式，都是這些基本的東西組裝而成。

從 scratch 來學這些基礎有兩個好處：1. 不被語法限制，2. 視覺化。scratch 裡面你幾乎不用打字，甚至可以看做是拼圖遊戲，你就把你想要的東西拼一拼，程式就寫完了。所以你學的是真正的「概念」，而不是語法。

舉例來說，如果你學 C，你就要學：

```
for(int i=0; i<n; i++){
  printf("hi\n");
}
```

如果你學 python，就會是：

```
for x in range(n):
  printf("hi\n")
```

（我不會寫 python，寫錯的話抱歉 QQ）

但這兩者本質上是一樣的東西，都是迴圈，只是語法不同而已！從 scratch 開始，可以讓你跳脫這些語法，只學那些精髓，你學完之後，看到 C 的這段就會知道：喔～這就是迴圈嘛。

第二點，視覺化也是很重要的一項。儘管我程式寫了十年，我永遠忘不掉，我在迴圈那邊卡了多久
尤其時雙重迴圈，根本就是惡夢！「我知道這邊是 1 到 10 的意思，但是第二層為什麼會這樣」、「為什麼這樣就可以印出九九乘法表？」

那時候我面對的是冷冰冰的程式碼，大概就是：

```
for(int i=1; i<=9; i++){
  for(int j=1; j<=9; j++){
    printf(...);
  }
}
```

儘管我現在來看，可以從排版跟大括號一眼看出「block」的概念，但十年前的我完全不行，完全不知道這段到底在幹嘛。可是 scratch 就不一樣了！它直接用視覺化的拼圖的概念，讓你在迴圈裡面可以塞另外一塊拼圖，就可以明白為什麼雙重迴圈會是這樣了！

最後附上我修某堂課的 scratch 作業，大概花了兩個小時，就只是一個簡單的小遊戲。

學著打指令（command line 操作）

「蛤？這是什麼？」就是電影裡面看起來很帥的，底是全黑的畫面，螢幕上面滿滿的文字，一張圖都沒有。在 windows 就是 cmd，命令提示字元；在 mac 就是 terminal，終端機。

假如你已經對上面介紹的 scratch 滿熟悉了，那你應該了解程式幾個基本概念，其中一個重要的概念是：寫程式其實就是在對電腦下指令。

其實呢，你平常在做的事情，也是在對電腦下指令，但作業系統幫你包裝好了，所以你只要動動手指就可以，例如說你在網頁上按了重新整理，「我要重新整理」，換成程式大概就是：page.refresh()；或是：「點了前往 google.com 的連結」window.open('google.com');

（這邊程式碼都隨便寫的，但意思有到就好）

我們所謂的 command line，就如同字面上這樣，是個「用文字對電腦下指令」的地方。像是 ls，就是 list 的意思，會把你「現在在的地方」的檔案都顯示出來，「現在在的地方是哪裡？」打 pwd，Print Working Directory，就會顯示出你現在在「電腦的哪個資料夾底下」

幫大家準備好幾個連結了：（關鍵字：終端機 教學）
〈OS X Lion 10.7 系統基礎教學系列 – 終端機基礎操作教學〉、〈介紹命令行 (command-line) 介面
Mac OS X Terminal 101：終端使用初級教程〉、〈Mac OS X Terminal 終端機常用語法教學〉

為什麼要學這個呢？因為工作上很大機會用的到，而且日常生活也很好用。有些功能你用 java 寫的要死要活，最後才發現原來電腦就有內建指令。我也是最近一年才開始熟悉這些指令的操作，才發現原來內建指令這麼好用。

我認為對初學者，你要知道「有這樣一個東西」，並且了解：cd, ls, pwd, touch, mkdir, rm, cat, grep 這些指令，還有 > >> < | 這些符號在幹嘛，這樣就很足夠了。

想更精進的可以去學怎麼寫 shell script

學怎麼用 browser dev tool

chrome、firefox、safari、edge 都有一樣的東西，叫做：開發者工具。

為什麼要學會這個呢？因為你可以了解很多東西！日常生活中，你其實一直在接觸網路的概念，只是你沒發覺而已。你平常看到的網頁，其實也只是一堆文字而已，那為什麼文字會變成畫面？因為瀏覽器幫你解析，按照一定的規則畫出來。你可以右鍵 -> 檢查，就可以看到你滑鼠指的地方本來的文字是什麼。

dev tool 可以講到一堆跟 web 有關的概念，get/post, ajax, status, css, js, html 等等，現成的好工具，不用嗎？

有興趣深入了解的，我會下的關鍵字：dev tool 入門，（只是目前應該很少文章是用 dev tool 教你學習網路概念，有點可惜）

選你想走的領域精進

假設你上面都有確實做到，你現在應該對程式、對網路都不會太陌生了，這時候再來學專門的東西，我認為是事半功倍。為什麼？因為原本那些程式基礎你都可以跳過，你要學的只是語法

學程式學的應該是「心法、內功」，而不是表面的限制，這樣就算換了一個你從沒看過的程式語言，你也會猜的到它在幹嘛。如果你這時候想要報名一些課程，那就去吧。

版上有很多心得可以參考，不過要注意的是心態要正確。我之前寫過一篇文〈那些「鹹魚翻身」的文章沒有告訴你的事〉。

有些新聞會打著什麼「服務生學程式，年薪暴增 300 萬」之類的，不要輕易相信這些，不要抱持著這些美好幻想。很多這種案例都是因為在學程式以前已經有某項專長，再把自己的專長跟程式結合起來，這些案例大多數時候並不適合「沒有任何背景」的人參考。

比起這些特殊的案例，版上很多文章都更有參考價值。版上很多心得文，看了就大概可以知道成效如何，出來的薪水如何。或是版上也很多徵才文，稍微看一下大概就能知道現在行情大概在哪裡。

寫文章記錄心得

可以開個 blog 寫寫自己的學習歷程、碰到的困難以及解法等等。

為什麼要寫這個？

第一是加深印象，你解決一個問題下次再碰到時，你還記得怎麼解嗎？我記憶力不太好，有時候過兩三個月會忘掉，但這時候就因為我有寫 blog 記錄起來，可以查的到以前自己怎麼解的。

第二是增進功力，我要怎麼知道我真的了解一個東西？我覺得寫下來，可以幫助你知道這點。例如說我今天要寫：git 入門教學。寫一寫發現自己不知道 rebase 在幹嘛，我就必須去查，查完之後寫在文章裡面，你寫的出來，你就一定懂這項技術。而且部落格是公開的，若有路人看到你寫錯，說不定還會來糾正你，你就又學到一課了！

第三是累積個人聲望，好的部落格對於求職絕對有加分效果。

假設你今天是面試官，A 的履歷寫：精通 Javascript，B 的履歷寫：精通 Javascript，可參照我寫過的：深入 Javascript 原理系列文章。結果你跑去 B 的部落格看，真的寫了一系列深入的文章在研究。不用面試，你就知道 B 是個有真材實料的人（前提是面試官會去看你 blog XDD）

學資料結構跟演算法

就是俗稱的 Data Structures and Algorithm，DSA

假如你真的對這行很有興趣，想要一直待在這邊而且薪水越來越高的話，你就勢必要開始補足一些「自學的人通常不會有的本科系知識」。

因為自學者通常都直接從 html, css, javascript, php, rails 等等的開始，甚至連資料結構跟演算法都沒聽過！有接觸而且實作過的更是少數。

那為什麼要學這兩樣東西呢？

第一，增加自信。其實能力強，自信就會強。增加自信是因為，自學者通常也是非本科系的學生，可能會覺得自己跟本科系的實例有一段落差。這時候若是學了本科系在學的 DSA，就可以彌補這一段落差。

以我的例子來說，我高中的時候就在接觸 DSA，打打比賽，寫一些 online judge 的題目（可參考：〈Re: [ 請益 ] 請問國中生程式設計競賽入門〉）。到現在出社會找工作，就是因為我有這段經歷。所以我有自信我絕對不會差本科系太多，甚至會比一些很混的畢業生強，事實上也是如此，去面試的時候有問公司考我排序的問題。就是一些很基本的，說你知道哪些排序法，跟時間複雜度之類的。我答完之後他們說：我是目前面試的十幾個裡面，回答的最清楚的（可參考：〈[ 心得 ] 15 家中小型公司、新創公司面試心得分享〉）

自信在某些時候很重要，能夠為你帶來「打不倒的勇氣」，面試沒上，沒自信的人會覺得：一定是我太爛。有自信的人會覺得：是公司沒眼光（前提是你的自信不是來自於自我感覺良好 XD）

第二，增進對電腦底層的理解度。剛開始你可能只是個拿 30k 的小螺絲釘，但工作久了，可能變成 40k, 50k，甚至 100k。當你薪水變得愈高，要解決的問題也相對的更重要，DSA 是大公司的必備技能之一，題目可參考 leetcode。

為什麼那些大公司面試要考這些？如果不會二元樹反轉，就算是知名工程師也沒用？（可參考：〈Google 面試這次惹爭議了：雖然我們公司 90% 的工程師都用你開發的工具，但我們還是不聘用你〉）

就如同我開頭所說的，重點在目的、在於要解決的問題。

你在一家普通的台灣電商網站工作，需要會什麼？你可能要會寫 php，因為要改進購物車，要改進物品上架系統，所以你根本不必懂 DSA，因為沒什麼太大的幫助

Google 要解決的問題是什麼？

可能是「排序 10 億個數字」、「阿發狗的下棋演算法」等等。從要解決的問題看來，你就知道為什麼 Google 面試時要考那些了。如果你要在技術上日漸精進，你遲早要碰到那些較底層的東西。

這跟問題規模有很大的關係，假設你公司今天是做售票網站的，同時 100 個人搶，一般工程師都可以自己應付。同時 10000 個人搶，這就不一樣了，在伺服器上你可以能要多開幾台機器，要調整一些架構，而程式當然也要跟著調整。（延伸閱讀：賣秒殺票跟你想得不一樣、號稱推不倒的售票網站 KKTIX，技術現況分享）

或是你會發現許多大公司，常常都會自己有一些專案。為什麼？因為市面上沒有符合他們需求的東西，所以要自己寫一個。像是 React 就是這樣，如果你沒修過 DSA，你怎麼可能實作出 virtual DOM diff 的演算法？（延伸閱讀：〈深度剖析：如何实现一个 Virtual DOM 算法 〉）

（以上對於大公司面試為什麼要考這些的那一段都只是我的推測，小弟本人沒那麼強，沒去面試過
如果有進去過的前輩願意跳出來說：才不是這樣！你根本推測錯了！麻煩不吝推文或站內信指正，感謝）

附上兩篇跟我講的內容有點關係的。〈Re: [ 討論 ] 所以練 acm 都底有啥好處？〉、〈過早最佳化是萬惡的根源〉

最後，舉 vgod 那篇裡面引用到的一段：

> 「如果以蓋大樓來形容這個概念，把砌磚作為一種技能，把蓋大樓作為一種知識。我想可以這麼講：如果你早就知道你喜歡砌磚，很會砌磚，就直接去砌磚吧。如果你的夢想是蓋大樓，你要學的東西還很多，那讀大學是你最好的途徑。不是每個人都要蓋大樓，靠砌磚就可以賺錢了，砌的好還可以賺很多錢，大家搶著要。」

如果你很會寫購物車，那其實在台灣你也可以接到很多電商客戶，錢包也是賺滿滿。但如果你是想進 Google、Facebook 那種大公司，或是想要思考規模更大的問題該如何解決，那就學 DSA 吧！

推薦這門 edx 上的課，對岸的清大的開放式課程

分享

這點呼應我前面提過的寫 blog。分享從來都不是單向的，在我分享經驗給其他人的時候，我也得到許多回饋。尤其是教學！在教別人的時候，你才會更了解自己哪邊不太懂

我認為分享是一件很重要的事，取之於社會，用之於社會。我從以前就常看 soft_job 版，學到很多。為什麼學的到東西？就是因為很多前輩願意 po 文分享自己的經歷。

因此我也決定哪天當我有能力，我也要出來 po 文分享，這樣才能有一個正向循環。我希望看我這篇文章有得到收穫的朋友，也可以記得這個道理，日後有心得記得在這裡 po 文分享 XD

推薦一些課程

我修過的線上課程其實不多，大多數都是零碎的片段知識，但我有一堂真的很推薦、很推薦的課：CS50。我寫的這篇文章有介紹這門課在幹嘛，因為篇幅有點長，所以就不貼上來了。但就是基本程式、底層、網路、資訊安全、資料結構都有教到！
如海洋般的程式課程：CS50

我上面說的：scratch, command line, dev tool 都是受到這門課的影響。他先教你 scratch，讓你理解程式基本概念，接著再教你 C。是的，初學者學 C 不太好，我原本也是這樣想。可是這門課幽默風趣，讓 C 不會這麼難，我有點改觀。

總之這門課很「硬」，很扎實，但也因為他扎實，我相信你修完之後，保證很有收穫。我自己都覺得我十年跌跌撞撞學的東西，怎麼他 20 幾堂課就教完了 XD

文章有點長，差不多該結尾了

我去年有在做免費程式教學，但較遺憾的是，其實大多數都是單純的問與答，比較少真的動手教程式。時間不太夠，我也不太曉得從哪個環節開始教。

這邊有我寫下的心得與記錄（台北免費程式教學心得分享），裡面我有寫了每個人的背景、問題以及我的回答，來找我的人大多都是新手，或許你可以參考看看。

最後，如果你真的想修 cs50 這門課的話。我想開個 slack 或是社團之類的，有興趣的可以一起討論。我全部作業都有乖乖寫，所以碰到問題我都可以幫忙回答 XD 有興趣的話可以寄站內信給我，記得附上 email

或是，雖然我不是什麼大大。但如果覺得我的文章對你有幫助，有些事情很迷惘想跟我進一步討論，也很歡迎直接寄站內信。感謝~ 對 cs50 有興趣的可以直接加入臉書討論區

另外，這幾年幫許多迷惘的初學者解惑，得出一點心得，規劃出一套我認為初學者很適合上的課程，能夠提升你的基礎知識與程式能力。目前正在 hahow 平台上面預購，有興趣的可以參考：初心者的計概與 coding 火球術。



延伸閱讀：

2025 年時代變化：軟體開發會成為地球上最後一項「有用」工作
我的自學路：克服安逸與惰性的唯一途徑，就在於把學習養成習慣
想看書自學軟體工程？就從這 6 本輕鬆書打造入門款概念