程式設計初學者之心智模式個案研究

A Case Study of the Mental Model on Novice Programmer

謝韶祁 [1] 賴阿福 [2] 楊政穎 [3]
HSIEH, SHAO CHI[1] LAI, AH FUR[2] YANG, CHENG YING[3]
[1] 臺北市立大學 資訊科學系碩士班 研究生
[1] Department of Computer Science, University of Taipei
E-mail：G10916015@go.utaipei.edu.tw
[2] 臺北市立大學 資訊科學系 教授
[2] Department of Computer Science, University of Taipei
E-mail：laiahfur@gmail.com
[3] 臺北市立大學 資訊科學系 教授
[3] Department of Computer Science, University of Taipei
E-mail：cyang@go.utaipei.edu.tw

摘要

本研究探討程式設計初學者，從學習程式前的想法到開始學習後之心智模式轉變過程，以及學習上所遇到的問題與解決方式，提供老師在未來教學、設計診斷評量、線上教材與系統時的依據參考。研究方法為透過深度訪談，針對一位程式設計初學之研究所學生進行學習研究。根據研究結果顯示，初學者 S 學生起初對於程式設計較缺乏自信，並且在學習上容易受到數學概念影響、程式語言基礎觀念的混淆等等而造成學習上的阻礙。但在經過課程由淺入深的教學，以及經由作業、考試與專題的製作，S 學生從過程中發現自己的問題，更重要的是透過反思性思考與學習歷程的撰寫以了解自我學狀況，並不斷從中改進學習，而 S 學生對於程式設計漸漸有了自信心，在學習上更是產生積極主動的心態。

關鍵字：程式設計初學者、個案研究、心智模式、反思性思考

Abstract

This research explores the process of novice programmer, from the idea before learning the program to the mental model transition after the start of learning, as well as the problems and solutions encountered in learning. It provides teachers with future teaching, design diagnostic evaluation, online teaching materials and The basis of the system. The research method is through in-depth interviews to conduct learning research on a graduate student who is a beginner in programming. According to the research results, beginner S students lacked confidence in programming at the beginning, and were susceptible to the influence of mathematical concepts, confusion

of basic concepts of programming language, etc., which caused learning obstacles. But after teaching from the shallower to the deeper, as well as through the production of homework, examinations and special topics, S students discover their own problems in the process, and more importantly, understand their self-study through reflective thinking and writing of the learning process. And continue to improve learning from it, and S students gradually have self-confidence in programming, and they have a proactive attitude in learning.

Keywords：Novice Programmer, Case Study, Mental model, Reflective Thinking

壹、 前言

在資訊科技日漸蓬勃的時代，人人都開始學習程式設計，然而基於每個人學習的出發點不盡相同，特別是剛踏入此領域的程式學習初學者，特別容易遇到學習上的問題。如何減低初學者們在學習上經常有的困惑，以更清楚穩固的學習打下扎實的知識基礎，是當前相當重要的課題之一。初學者若能在初次學習的程式語言就建立下良好的基礎，不僅在培養邏輯思維之概念上將有所提升，在日後不管是再深入學習其他程式語言，或是學習其他新知、做事處理上，也將會有更好的表現。而本研究將進行質性訪談之個案研究，分析過去從未接觸過程式設計的初學者學生，了解其在學習程式設計時經常遇到的問題，以及觀念上容易迷糊的地方，並透過分析、整理出學生在學習上常會遇到的盲點區、易發生錯誤與混淆的地方，提供教師在教學上應多注意的部分與依據，以及設計線上教材、學習系統、診斷評量之參考，以減少剛跨入程式設計學習的初學者所會遇到的阻礙問題。

貳、 文獻探討

一、程式設計學習與初學者

(一)程式設計學習

隨著世界的潮流與教育政策的改變，許多國家早已將程式設計加入至課程當中 (Balanskat & Engelhardt, 2015)，台灣的教學課綱中近年來也將「程式設計教育」加入必修，程式設計融入到課程教學中可說是越來越廣泛普及。

學習程式設計不僅僅只是學習如何撰寫程式，一方面主要透過程式設計的學習與編寫思考過程，提高學生解決問題、推理、邏輯思維及創造力等等的能力 (Moreno & Robles, 2016)。Scherer 則指出程式設計可以提升學生再創造、解決問題等關鍵技能的表現 (Scherer, 2016)。Marimuthu 與 Govender 也表示程式設計在近年來已經變成相當重要的技能，可以培養解決問題的能力以及邏輯分析推理能力 (Marimuthu & Govender, 2018)。透過以上文獻描述顯示，程式設計在這個數位化時代已經相當重要，而透過學習程式設計其背後所培養的能力，如解決問題

能力、思考力及邏輯能力等更是重要。

（二）程式設計初學者的困難點

程式設計對於初學者來說，是個較於複雜且不容易理解的過程，並且有一定的困擾與難題(陳美文，2018)。而學生通常認為學習程式設計是困難、無聊又複雜的過程，因而減低學習的動力(Papadakis , Kalogiannakis , Orfanakis & Zaranis, 2014)。也有研究指出，學生認為程式設計是難以理解的過程，主要因為編寫工具和程式結構的規則語法(Erümit , Karal , Şahin , Aksoy , Gencan,& Benzer, 2018)，有以上可知，程式設計對於初學者來說，是有著一定的難度與挑戰。

而在學習程式時遇到問題是相當常見的事情，如學生在程式設計就時常常會遇到很多的問題，但若沒有清楚的解決遇到的問題，將會增加其日後學習的困難(Durak, 2020)。很多時候只考慮到計算機科學的發展，但卻忽略了程式初學者的學習狀況，而讓初學者在學習曲線上顯得不親近 (Schwartz , Stagner, &Morrison, 2006)，因此造成很多的程式初學者會因為起初所遇到的困難，而誤以為自己沒有天分，不適合學習程式設計，繼而降低學習動機甚至到最後放棄。

二、邏輯運算思維

「運算思維」是資訊科學教育中相當重要的核心，運算思維並非單指程式設計，並也代表了一種思考方式，為程式設計中相當重要的新能力(劉晨鐘， 2019)。學習程式不僅僅是學習程式如何設計，而是透過學習程式的過程培養邏輯思維等的能力。運算思維在這個時代中可說是不可或缺的技能，影響了一個人的批判思考、解決問題及想法思維 (Yünkül, Durak, Çankaya, & Mısırlı, 2017)。然而對於許多教育者，在教導學生程式設計技巧和邏輯思維的過程中遇到許多困難(Kalelioglu, 2014)。透過不斷的練習，學習程式的過程中也培養著學生的運算思維，當學生的邏輯思路越來越清楚時，也將能在程式設計的過程中更順利。邏輯運算思維對於程式設計初學者可說是相當重要的一環。

三、心智模式

心智模式的概念已經存在已久，在 1943 年時就被 Craik 所提出。心理學家 Johnson-Laird(1983)認為，心智模式是人們理解日常所發生的現象，並加以進行推論預測，解決問題時的過程。Norman(1988)則表示，心智模式是我們對於事件的發生、行為的方式的概念模型，是由我們對於事物的解釋而形成的，而對於了解經驗，預測行動、處理事件相當重要。心智模式是心智實體的組合，首先要與事物有所互動，並反映事物的相關狀況 (張志康、林靜雯、邱美虹，2009)。心智模式會隨著時間的推進，並持續受到外界的事件而發生改變 (陳怡靜、胡學誠，2012)。而心智模式在學生的學習上，也影響了其在學習程式中的課堂表現(Ramalingam, LaBelle, & Wiedenbeck, 2004)。

綜合以上整理，心智模式是根據個人與外在事物互動後所產生的看法，其中會影響心智模式原因包括個人思維、外在事件影響等，而也會影響之後面對事情

時的思考應對，並且會隨著時間的推移而有所改變，而本研究將透過了解程式設計初學者的心智模式，深度的了解其對於程式設計的學習前後之心路歷程與感受。

四、反思性思考

在教育領域裡，反思性思考這個理念很早就出現，特別在專業教育中日漸受到重視 (Lim & Angelique, 2011)。反思性思考是一種技巧，可以幫助發現當前的學習狀況，並針對遇到的問題去制定解決策略與改進過程 (Kizilkaya & Askar, 2009)。而反思性思考只有在遇到某問題時才會出現，是學習時相當重要的一個過程(Durak & Yilmaz, 2019)。Phan(2007)指出，反思性思考在教學過程中是有幫助的，使學生和教育工作者都可以對自己的學習或專業進行批判性思考，幫助其在專業領域內發展專門的知識。反思性學習在課程學習中是相當重要的一部分(Bourner, 2003)。Chen、Hwang 和 Chang(2019)採用反省性思考提升策略，進行「數位學習策略分析與應用」研究，研究顯示反省性思考提升策略能顯著強化學生學習設計專案產出、反思性思考以及課前參與和投入程度。

透過反思性思考，學生可以更釐清自身所遇到的問題，教師也透過學生的反思性反饋，更助於理解學生的學習狀況與所遇到的問題(Schaaf, Baartman, Prins, Oosterbaan & Schaap, 2013)。綜合上述文獻，反思性思考對於學習有相當大的重要性，一方面可以更清楚並評估當前的學習、了解自己心中的想法，一方面透過反思對於問題進行思考，讓自己印象更深刻、了解其意義，並發覺自己可能不足的地方，進一步的學習與改進，因此反思性思考在學習上是相當重要的一部分，因本研究也將深入了解程式初學者在反思性思考這方面的感受過程。

參、研究實施與設計

一、 研究方法

本研究採個案研究，以質性訪談的方式蒐集資料，針對一位程式設計初學者（以下稱為 S 學生）進行訪談，在訪談的時間上，將訪談分成兩次進行，分別為學期初與學期末兩個不同階段，以收集 S 學生在初學程式的歷程情形，目標希望經由兩個不同時間，配合學校程式教學的進度，可以更清楚的收集到 S 學生在不同學習階段的經驗與過程。

質性研究中之信效度，使用 Denzin(1978)所提出的三角檢驗，主要分為四種鑑定模式，包括：(一)資料來源的不同的來源三角鑑定;(二)一位以上研究者進行研究的分析者三角鑑定；(三) 透過多種理論觀點進行研究的理論觀點三角鑑定；(四) 以不同資料蒐集法檢驗研究，檢驗研究發現一致性的方法三角鑑定。而本研究透過使用三角檢驗中的「分析者三角鑑定」，除了研究者本身對於資料的詮釋解讀，另外請兩位專精於數位學習領域的教授一同對資料進行分析，最後進行比對與校正結果。

二、 研究工具

目標深入的了解S學生在學習程式設計的狀況,將透過深度訪談中的半結構模式,先在訪談前根據研究的問題目的擬定相關訪談大綱,並交給數位學習領域教授進行檢閱修改。並在正式訪談時根據擬定好的大綱進行訪談,如過程中若發覺有較特殊、印象深刻的經驗或是適合更深入探討了解的內容時,將依實際狀況彈性調整訪談內容,進而收集到更豐富完整的資料內容。

在訪談過程中,因最重要的資料為訪談時的相關內容,因此為了避免訪談內容的問與答有所遺漏,全程將使用錄音的方式記錄,並隨後將訪談之內容轉譯成逐字稿形式,再進行編碼整理與分析,以了解S學生在初學程式設計時其心智模式以及學習情況。訪談大綱題目如下:(一)請問學習程式設計的動機為何?(二)學習程式設計前的感受為何?(三) 學習程式的情形如何? 過程中常遇到甚麼困難、比較難懂的地方?(四) 程式中哪些觀念容易混淆?怎麼解決?(五)有沒有學習不一樣的程式語言? 有甚麼想法嗎?(六)初學程式及至今的學習感受與改變?

三、 研究個案描述

本研究將訪談一位目前正就讀北部資訊類研究所的S學生,而此學生過去無程式設計相關經驗與基礎,在學歷上目前已有其他非資訊相關科系之碩士學歷。然而S學生起初會選擇再去修習資訊研究所之原因,主要因為在過去學習的過程中,一直不確定自己的興趣與未來目標,而後來決定以就業取向為選擇科系類型之考量方向為出發點。S學生認為在資訊領域中的工作發展較多元,因此選擇再進修資訊類研究所,希望在就讀的過程中,進行資訊領域的學習,並練習程式設計,增加自我專業能力。

S學生在程式語言的學習上,於碩一上學期時修習JavaScript課程,而碩一下學期修讀Java和App Inventor課程,而本研究將透過訪談,針對S學生目前學習到的程式語言之學習情況進行訪談,目標希望透過訪談,深度的了解S學生在學習程式時,心智模式發展的過程與轉變,以及對於學習程式設計的感受、態度與學習歷程經驗。

肆、 研究結果

根據深度訪談的內容結果,將程式設計初學者學習歷程與心智模式轉變過程進行整理如下:

一、初學者對程式設計之學習情形

(一) 學生初次學習程式設計之初學感受

1. 學生學習程式設計動機

S學生在過去高中與大學時期,因為課程上的安排緣故,未曾接觸程式設計

相關課程。後來則因為考量到未來就業，資訊相關發展較多元，而選擇了再去進修資訊類的研究所，開啟學習程式設計的動機。

「起初在高中的時候，完全沒有接觸過程式，因此對程式設計也完全沒有概念。」

「在讀完大學時快畢業時，才真正開始認真思考以後的工作要做甚麼? 而想了又想，甚至上網查詢相關資料，發覺如果工作後走資訊相關，工作發展出路會比較多元，且比較好找工作。」

2.初學程式設計之感受

起初S學生認為學習程式設計是有一定難度的技術門檻，並對自己較沒有自信心，其中一大原因是因為對程式沒有概念，而導致程式設計對S學生來說，在心理層面上是有壓力的。在實際開始學習之後，透過課程由簡單的基本概念開始學習，建立初期的自信心，進而發覺程式設計並沒有想像中那麼難。S學生表示雖然在學習上、作業上會遇到困難，但解決問題之後，程式可以運作時就感到很有成就感，而這也是 S 學生持續學習下去的動力之一，不再對程式設計感到畏懼。

「只是心裡對程式設計的第一感覺，是一個有難度的東西。」

「而最一開始學習程式時，才發現沒有想像中那麼難，但也才發現原來程式有著這麼多的規則需要遵守，而在實際操作上，像是在做作業時，有時雖然要做很久，甚至常常遇到找不到問題而煩惱，但是在最後成功做出來且可以執行時，就會感覺很有成就感，雖然不是說多厲害的程式，但是覺得自己又多學習到了一些東西技巧。」

在剛開始學習寫程式的過程中，S學生常發生有設計想法，卻不知道該從何開始的情況，或是不知道如何將腦中的想法轉化為程式的樣貌。這部分推測是因為在程式的基礎還並未建立起來的關係，並且在程式的活用上還未熟悉，因此會有思考上容易遇到瓶頸的情況。

「常常在寫程式時，腦中雖然知道可以這樣操作而得到結果，但是實際上要開始執行時，卻會發生不知道從何做起的狀況，導致常常會卡住。」

(二) 學生經過程式設計學習後感受

1.透過作業與考試發現自己不足的地方

S學生表示，透過課堂的練習，包括課前小考和課後作業等等的練習，可以很快的發現自己還未熟習、觀念上有理解錯誤的地方。而S學生因為是程式初學者，常常只要題目有一些變化或是小陷阱就會很容易做錯，但也是因為有這樣的機會，才會讓學生透過反覆性的思考，找出自我不足的地方，並加以修正。

「透過課堂上老師的教學，以及每次作業與小考的練習，一次次都會遇到一些自己不懂的地方，或是觀念錯誤的地方，雖然常常遇到錯誤，但覺的每一次的學習都讓自己更進步一些，這次釐清觀念之後，再下一次就會更清楚，慢慢的腦中也會對程式越來越有印象，知

道哪邊應該要注意才不會出問題。」

在撰寫程式時，S 學生有時會遇到不知道該如何設計程式以處理題目需求，而程式設計的過程，需要對於題目所需進行程式的演算法設計、程式分析與資料剖析等等的思考與安排，而這些部份與邏輯運算思維有相當大的關係，包含批判、思考與解決問題的能力。透過每次的程式設計實作練習，就在訓練邏輯思維，當邏輯越來越清楚時，在設計程式時也就會更順利。而 S 學生表示，像是有一次遇到需要將文字反轉輸出的題目要求，然而腦中雖然有想法，卻不知道該從何下手設計的情況，而這部分在經過多次的練習，不斷的培養邏輯思維，就越來越能掌握題目要求並設計適合的程式。

「在寫程式時，有些時候會卡住，有時是因為對於題目所要求的東西，我想不太到該怎麼處理才適合，有時則是雖然大概知道該怎麼做，但在程式內部該怎麼轉換與設計卻有些迷糊不清楚，而造成卡在這個問題當中，像是在之前有遇到一個題目，要將使用者輸入的文字反轉輸出，然而我雖然有想到可以利用迴圈的方式做處理，但是實際上要怎麼設計程式，我卻想不出來。」

「雖然會遇到不會做的題目，但是我覺得經過每次的實作程式設計，都在幫助自己學習與釐清觀念，也培養對程式設計的敏銳度、累積更多的經驗和培養思考的方式，而當往後遇到更多不同程式問題要解決時，在邏輯上會更加清楚，想法思路也更清晰。」

2. 不同程式語言相似處

S 學生在上學期學習了 JavaScript 程式語言，而在這學期學習了 Java 和 App Inventor。學生藉由透過學習不同的程式語言，逐漸了解到各種不同程式語言之相似處，如基本的變數使用前須宣告，或是 for 迴圈、條件判斷 if else 的使用等等，在基本概念架構上都相同，也因為在各不同程式語言進行學習與比較，讓 S 學生對於程式語言基本概念更加理解，也在學習程式語言時更加有自信，相信自己可以透過努力學習起來，並與剛開始學習程式時增添許多信心，也對後續的程式學習更加有幫助。

「在上學期學了 JavaScript，而現在學習 Java 和 App Inventor，學習的過程中，漸漸地發現有一些地方很相似，像是使用變數前要先定義，或是做一些判斷迴圈 for、if else，發現這些地方其實在使用上面的觀念都相當相似，只是在呈現上以及語法上有一點點不同。」

「而透過不同語言的學習，讓我更熟悉且清楚知道程式的用法。而目標要達成一樣的事情，在這樣的比較與實作當中，對程式語法有所深刻，並清楚的可以知道原來都大同小異，對學習也更有方向與動力，知道學好一種語言之後，其他語言就會很快的可以融會貫通。」

3. 改變使用暴力破解的方式寫程式

起初剛開始學習程式的 S 學生，其在編寫程式時常常只會想著把程式寫出來並能運作就可以了。像是學生在製作亂數出題不重複的陣列索引值時，因為對於 for 迴圈的觀念與運用還不是很熟悉，因此另外使用了較為辛苦的方式，一個一

個將值填入陣列中再去打亂達到亂數不重複，然而這樣的方式雖然可以達到一樣效果，但是若在之後要更改數量時，就不利於後續的維護修改。另外 S 學生在解決之前的一項作業之問答測驗其中的計分程式設計時，也使用了較直接的暴力破解，以完成老師所要求之每答錯一次就少加 2 分的計分方式。由此可發現程式初學者因為會的技巧較少，因此會使用較直覺的方式撰寫程式。

「起初在學習程式時，只想著能把程式寫出來就足夠了，因為所會的技巧還不多，也對方法還沒有很熟悉，所以寫出來的程式通常都較長。」

「常常看到一個題目就先一股腦的直接先做了，想到甚麼就寫甚麼，當然這樣子寫出來的程式我也不覺得哪裡有問題，也可以成功執行。」

「像是在學 javascript 的時候，有一個作業要做亂數出題，會用到是要自行建立陣列，再來把數值放進去陣列中再打亂，而還不太會活用 for 回圈，就直接用手動的方式一個一個給值，但是缺點是只要輸入很多次，而且只要數量一改變又要重新輸入，花很多時間。」

「在有一次作業中，要做題目問答測驗，若使用者一次答對就加 5 分，而此題每答錯一次就少加 2 分，直到 5 分被扣完為止才換下一題繼續。而因為我只想到較直接的做法，因此直接的設計程式在第二次答對時加 3 分，第三次才答對則只加 1 分的解法，而這樣的方式雖然有達到老師所要求的，但是發現這樣的方式不太聰明。」

但是在學習了一個多學期的程式語言後，隨著學到的技巧、看過的程式越來越多，開始會思考除了寫出來之外，是否有更好的方法可以讓程式更精簡聰明且有效率，日後維護上也更加輕鬆。

「但是隨著課堂慢慢的在進行，老師所教的東西越來越多且複雜，這時雖然一樣可以用很值白的方式去寫程式，但是常常程式會變得相當龐大，而很多地方重複在作一樣的事情，但是透過老師的教學和同學間的討論，我才開始了解可以換個角度去思考，用更聰明的方式去編寫程式。」

4. 學習後的心得

從起初沒有程式設計的概念底子，而對學習程式有所擔心的 S 學生，在跟著學校課程從基礎開始學習，同儕之間的互助學習，漸漸地對於程式這領域越來越熟悉，在 debug 能力上也越來越進步，透過反覆的練習之下，一方面在自信心有所提升，而學習時更積極，對於未來在持續學習程式上的心態更加堅定。

「從一開始我對程式相當的不熟悉，在語法上和規則都花了多時間學習，而跟著學校教學課程一步步的學習，並和同學間互相討論學習，開始可以邊寫出一些程式，過但程中常常會遇到很多困難，尤其在 debug 實尤其需要耐心，有時候只是一個小錯誤就會導致整個程式不能運作。」

「到現在我比較能抓的到程式的概念，雖然對於太複雜的目前還是不太行，但是我相信持續學習，打下穩定基礎，在程式設計上會更進步。」

S 學生表示在目前學習的程式語言當中，以 Java 的規定最為嚴謹，也因為如

此，在學習時常常會遇到預料之外的錯誤。但是 S 學生相信只要將這些基礎規則好好的去學習，奠定了扎實的基礎，日後再學習其他程式語言時，可以更快的熟悉上手。

「雖然說 Java 規定相當多，而常常用錯方法，但是經由這樣的訓練下，讓我更明白程式的每一步在做甚麼，為什麼要這要規定、設定等，反而讓我在基礎中學習更多，雖然說還是在學習很基本的東西，但是一次次的上課與練習下，還是覺得有在進步，甚至以後在學其他語言時，也可以更快的上手。」

二、初學者對於程式設計遇到的困難與處理方式

（一）透過考試與作業發現自己不足的地方

經由課堂要求之大大小小的考試與作業，S 學生表示經由這樣的學習過程中，較容易找到自己觀念不清楚的地方。考試主要是老師了解學生是否真的有徹底瞭解這部分的觀念，而同時也會將題目做一些變化，因此只要觀念還不熟悉，就容易作答錯誤。作業則是讓學生透過實作，一方面複習上課所教的進度，一方面在實作的同時，讓學習到的知識在腦中留下更深刻的印象，並且了解每個步驟的意義何在，並學習如何活用。S 學生表示在這樣的安排下，學習的更確實完整，觀念也更加清楚。

「透過考試很常會找到出自己沒有很清楚的觀念或理解錯誤的地方。」

「有時在課堂上老師會出一些程式輸出的考試(會有迴圈、while 等等)，以了解我們的思考邏輯是不是有問題，而通常這樣的考試老師經常會放一些陷阱在題目當中。」

「因為透過這樣的考試，常常會發現自己以為自己懂，但其實不是徹底了解的地方，透過反覆的檢討錯誤點，讓我在思考的思路上越來越清楚，也更知道程式的規則有哪些。」

「這部分也是在經過每次老師上課前的小考，透過題目反覆的去練習，才能看出自己的問題點在哪裡。」

（二）反思與學習歷程的重要性

在 S 學生學習程式的過程中，持續的有在撰寫反思學習歷程，在經由訪談中發現，起初是學校老師要求須將遇到的問題或是解題過程想法進行記錄。而 S 學生發現透過撰寫學習歷程的紀錄，可以更清楚的了解當下自身學習狀況，並反思學習過程中、實作過程中遇到甚麼問題、如何去解決，像是 S 學生對 for 迴圈和陣列的使用上還不是很靈活，因此將遇到的相關問題都記錄起來。

「起初雖然我不太了解為什麼要多花時間打這樣的紀錄，但隨著課堂，一次次的做這樣的學習歷程反思，我開始漸漸瞭解到這樣的記錄對自己的好處在哪。在每一次寫學習歷程反思時，腦中就會開始思考自己目前在做題目時或做作業時、甚至是上課時，遇到了甚麼困難。」

「在寫了學習歷程後，了解到自己對 for 迴圈的格式還沒有很清楚，導致只要題目稍微有一

點陷阱，我就會做錯，也有像是陣列的取值與顯是不太會使用，我也把這部份的解說寫在歷程當中，這樣的話就可以快速的去複習並加深印象，遇到類似的問題時也方便自己去做比較。」

S學生也表示因為經過這樣的反思，檢視、釐清問題並處理問題，有助於學習上的進步。日後若是想再次檢視複習或是又遇到相同問題，也透過反思性學習的紀錄回憶。

「我會馬上的先把問題記下來(學習歷程)，並盡量馬上搞懂，不然拖太久就會忘記，而且下一次的考試若是出了相似的題目還是不會。」

「這樣的反思讓我對我所做的事情又更深刻的複習了一次，也記錄下來供日後可以查閱複習。甚至是老師上課時哪裡我不太清楚，我就會做了紀錄，並且將得到的解答結果也記錄起來)。或是要問問題時，他人也可以依照我寫的學習歷程，更快且清楚的了解我的問題在哪。」

「我會將自己錯誤理解的縮寫紀錄在學習歷程中，把自己為什麼會看錯的過程寫出來，並加以註記告訴自己怎樣想才是正確的，這樣以後如果遇到類似的題目而不確定時，就可以很快的參考自己所寫的反思歷程，這樣一來才能將上次的問題與這次的做一個總和，而對這塊更了解。」

(三) 透過多撰寫程式學習並增加印象

在初學者學習程式設計的過程，因為基礎都還在建立當中，對於程式語法規範等還並不熟悉，在這樣的情況下編寫程式也容易遇到錯誤，或是不知道從何開始進行的窘境。而對於S學生，主要透過多加練習的方式，從錯誤中學習，這樣一次次累積下來，進步的幅度相當可觀，並也在腦中留下較為深刻的印象。

「雖然目前了解這樣的道理，但還是要多練習各不同的變化題目，才能再找出自己的盲點。」

「這部分隨著課堂操作和作業的撰寫，都會用到 cmd 去執行，而到現在我都會先看程式說第幾行出錯，再試著看 cmd 所說的錯誤內容是甚麼(試著解讀)，並開始除錯，而一次次這樣的除錯過程讓我對一些較簡單的除錯比較看的懂，而知道該怎麼去解決，我覺得多練習是相當重要的環節。」

「在後來自己實際多寫幾次程式後，就比較清楚語法上的用法，我認為是因為剛開始對程式規則不夠熟悉，所以在編寫時很容易就照以前既有的想法去編寫，導致會不斷發生錯誤，因此練習是進步的不二法門，多做才能發現錯誤並釐清觀念，對於我這樣的初學者相當重要。」

在學習程式時，總是會有某些較難理解、容易犯錯或忘記的部分。而初學者S學生，本身也經常會遇到這樣的情況，其主要以反思學習紀錄與練習為主，當問題是過去有發生過的，就會先去觀看反思學習紀錄，檢視當時的自己是如何解決問題，進而加強錯誤不熟的部分，複習過去所遇到問題錯誤。或是當過了一段時間沒練習某些程式技巧時，S學生也表示會有忘記的情況，因此為了減少這樣

的情況，也會盡量當有機會時就實際動手編寫程式，隨時當遇到問題就紀錄並想辦法解決，久而久之就會印象深刻。

「在學習的過程中，有時可能搞懂了一個東西，當下知道怎麼去使用(或是知道哪裡做錯)，但是只要過一個禮拜兩個禮拜沒再去複習，我就會又忘記。因此我覺得做好反思紀錄並經常的去練習，這樣的話就可以快速地複習，記錄之前錯誤的地方概念、易混淆的地方做複習，並再去學習新的東西，而不會學了這個又忘了之前的東西。」

「最主要還是實際去編寫與練習可以讓自己加強錯誤的地方，並多多撰寫學習歷程，讓每一次自己的進步都看的到。」

(四) 透過製作專題練習與學習

除了平時的作業之外，期末的小型的專題作業能促進學生思考，將過去所學做一個整合運用，不僅考驗著是否有將課堂教的知識學習起來，也考驗著學生是否能活用將所學轉化成自己的東西。S學生表示透過專題的製作，再次的熟悉了整學習課堂所學到的東西，也因此找到一些不熟悉的部分，但是透過與同學互相教導學習下，有時會有額外的收穫，印象也更深刻。

「透過課堂製作這樣的專題，而需要綜合了上課中所教的所有東西，而且實際的撰寫內容，主要在做專題的過程，我因此更熟悉了課堂所教的東西，也因為實際去做而會在過程中發現自己不熟悉的地方，或是搞錯觀念的地方，因此我覺得親自開始動手做是個在學習中滿重要一部分，雖然我常常會有想做卻做不出來的東西，但是可以跟老師和同學討論而學習到原本沒有想到的方法，而有意外的收穫，我覺得感覺很好也很踏實。而且才會讓自己學習更全面的感覺，而且腦中的印象更會更深刻。」

三、初學者學習程式時容易混淆的部分與觀念

根據S學生目前的學習進度，經由深度訪談的內容整理，將初學者S學生在學習程式設計容易遇到的錯誤進行整理，最終分類出三大類別：

(一)被既有數學概念所影響

對於初學者來說，受到過去習慣之數學思考模式，在很多時候會不經意的將數學所用到的符號等等放到程式當中，而導致在編寫完，程式開始執行時遇到一些問題。如大於、小於和不等於等等運算子的運用、也有像是賦值運算符(=)與比較運算符(==)混淆的情況發生。

「因為在以前讀數學時，只有一個等號(做比較用)，而一個等號和兩個等號我一開始學習時常常會弄錯。」

「常常在編寫程式時，腦中會對(比較大小的式子)出現混亂，比如<=和<u>≤</u>的式子判斷之語法，因為自己對程式的語法還不是說非常的熟悉，有時會一個不注意就將過去寫數學的習慣(如:不等於≠，!=)等等用法搞混，導致程式在實際執行時就會遇到錯誤。」

「當在撰寫比較範圍的程式時， 一開始寫這樣(90>= score>=100)，發現不能編譯，而分開之

後，(score>=90 && score<=100)這樣撰寫後才能執行。」

(二)程式縮寫

程式當中時常會用到縮寫的表達方式，讓程式可以更精簡，如三元運算值的表示法、++i 與 i++的差別等等，然而初學者在這部分因為還不熟悉，因此當程式有一點變化時，就會看不懂縮寫部分要表達的意義，而判斷錯誤。

「在學習的過程，偶爾會看到範例程式碼使用縮寫表示，而當有這樣的情況時，自己很容易會誤會其中的意思。」

「老師在教 Javascript 時，講到了一個詞” 三元運算值”，那時老師上課有做比較(就是和一般 if else 做比較)，相比之下很簡潔。原本以為懂了的我，當在課堂小考時，其中一題考了有一點變化的三元運算值(有包好幾層)，才發現自己不是真的懂，而發生不知道應該從哪裡開始看。」

(三)語法基本觀念

在程式語言當中，包含許多的語法與觀念在裡面，然而因為初學者在基本觀念還不夠熟悉，因此在設計程式時很容易就會有誤用的情況產生，導致編寫出邏輯錯誤的程式。如 S 學生所遇到之使用 for 迴圈狀況中，包括程式碼執行之順序判斷、巢狀迴圈結構或變數作用域等等，常發生誤用或不會用的情況。甚至當多種語法如陣列、運算子、if else 都混在一起使用時，S 學生表示這樣的狀況最常讓自己混淆，而不知道怎麼進行。

「這些變數是不是能活在迴圈 for 中?還是說出了迴圈之後這些變數就沒用了，這些地方我很容易搞錯。」

「當 for switchcase 三元運算值 if else 都混合在一起時，判斷常錯誤。」

「當多個程式語法題目混在一起時，常常會腦袋很亂，不知道哪個要先做。哪個是後面才做，而如果又加上很多層，常常在課堂上的小考時，就會有點慌，不知道該從何開始下手，只能硬著頭皮找比較簡單的題目先做。」

「for 迴圈的執行順序起初我常常弄亂，像是(b=1;b<8;b++)初始值甚麼時候要判斷有沒有符合，並在持續做下去，有時就因為複雜一點，而常常讓 for 迴圈少做一次，而造成判斷錯誤。」

「兩個 for 迴圈疊起來時，在思考執行上的順序時就亂掉，不知道是第一個迴圈執行完之後才再執行裡面的 for 迴圈，或是外面的 for 迴圈執行一次後，再繼續將裡面的迴圈跑完再跳出來繼續。」

S 學生在學習程式語言 Java 時，因為此語言重規則且嚴謹，因此在學習與設計程式的過程中更需注意許多細節與規範技巧，如目前會遇到如資料型態與轉換、變數命名規範使用、運算子設計等等。在透過不斷重複的練習之下，也培養著邏輯思考力和概念基礎的部分，而當自己掌握了一定程式設計原則時，不管在

日後學習或遇到更複雜的需求技術時，也才能更快的進入狀況，以及聰明靈活的設計程式。

「像有一次考試有出三個等號一個變數的指派，p=q=i=60 像是這樣的給值方法，我一開始並不知道可以這樣宣告，會猜想說是不是無效宣告。」

「因為 Java 的規定很嚴格，因此對於剛學習此語言的我，很容易會忘記少打東西，如沒有先定義變數再使用、double float 忘記使用，或是沒注意到規則而發生錯誤，如最後沒有分號、if else 沒有加大括號、字母大小寫，甚至是錯誤語法使用。」

伍、結論與未來展望

根據研究結果顯示，初學者 S 學生在學習程式的過程中，從起初對程式設計學習感到沒自信，然而透過由淺入深的課程教學下，慢慢的建立自信心，甚至從中感受到自己的進步狀況，而更相信自己能學習好程式。課堂透過考試、作業與專題製作，實際的讓學生動腦並撰寫程式，將所學到的知識試著做使用，一方面學生會更熟悉當前所教的東西，也會在過程中發現自己不熟悉的地方。

初學者在學習的過程中經常會出現許多迷思與不懂的地方，如當程式一變化，或是多種概念混在一起使用時，更容易發生混淆的情況，同時初學者也常受到過去學數學的符號與概念影響、或是在撰寫程式時因為對程式熟悉度不足而使用暴力解法。這時重要的不僅僅是讓學生了解正確的方法與觀念，更重要的是促進學生在學習後的反思性思考，透過每次反思與學習歷程的紀錄，可以幫助學生思考自己是否是徹底了解概念，並發現自己不足、可以更好的地方，也再次的複習所學，若日後發生類似問題時，可以馬上的檢視並記取問題所在。

綜合以上，程式設計對於初學者來說，是有一定難度的學習過程。本研究整理 S 學生從完全沒有學過程式到開始學習程式之心智模式轉變過程，以及學習過程中所遇到的問題迷思與困難。首先提供教師當作未來在程式設計教學上，如何教導學生的一個依據，包含迷思概念、不正確的心智模式等，並且作為發展診斷評量教學的參考，也可以依據研究結果，設計線上教材及模擬系統開發。

參考文獻

一、 中文部分

張志康、林靜雯、邱美虹(2009)。跨年級中學生串並聯電路心智模式的研究。**科學教育月刊，31**(3)，2-17。

陳怡靜、胡學誠(2012)。大陸學生選擇來台就學之心智模式研究。**創新與管理，9**，111-144。

陳美文(2018)。**VB 程式設計初學者除錯行為分析**。國立臺灣師範大學資訊教育研究所，台北市。

劉晨鐘(2019)。運算思維與程式設計教育浪潮。**人文與社會科學簡訊，20**(4)，89-92。

二、 英文部分

Balanskat, A., & Engelhardt, K. (2015). Computing our future: Computer programming and coding - Priorities, school curricula and initiatives.

Bourner, T. (2003). Assessing reflectivelearning. *Education & Training, 45*, 267-272.

Craik, K. J. W. (1943). The nature of explanation. Cambridge, UK: Cambridge University Press.

Chen, M. R. A., Hwang, G. J., & Chang, Y. Y. (2019). A reflective thinking- promoting approach to enhancing graduate students' flipped learning engagement, participation behaviors, reflective thinking and project learning outcomes. *British Journal of Educational Technology, 50*(5), 2288-2307.

Durak, H. Y. (2020). The Effects of Using Different Tools in Programming Teaching of Secondary School Students on Engagement, Computational Thinking and Reflective Thinking Skills for Problem Solving. *Technology, Knowledge and Learning,25*(1).

Durak, H. Y., Yilmaz, F. G. K., & Yilmaz, R. (2019). Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. *Contemporary Educational Technology, 10*(2), 173-197.

Denzin, N. K. (1978). The research act: A theoretical introduction to sociological method. New York: McGraw-Hill.

Erümit , A., Karal , H., Şahin , G., Aksoy ,D., Gencan, A., & Benzer, A. (2018). A Model Suggested for Programming Teaching: Programming in Seven. Education and Science, 1-29.

Johnson-Laird, P. N. (1983). Mental models - Towards a cognitive science of language, inference and consciousness. Cambridge, MA: Harvard University Press.

Kalelioğlu, F., & Gülbahar, Y .(2014) The effects of teaching programming via scratch on problem solving skills: a discussion from learners' perspective. *Informatics in Education 13*(1) , 33–50.

Kizilkaya, G., & Askar, P. (2009). The development of a reflective thinking skill scale towards. *Education and Science, 34*(154), 82-92.

Moreno, J., & Robles, G. (2016). Code to learn with Scratch?A systematic literature review. *IEEE Global Engineering Education Conference*, 150 - 156.

Marimuthu, M., & Govender, P. (2018). Perceptions of Scratch programming among secondary school students in KwaZulu-Natal, South Africa. *The African Journal of Information and Communication, 21*, 51–80.

Norman, D. A. (1988). The design of everyday things. New York, NY: Doubleday.

Phan, H. (2007). An Examination of Reflective Thinking, Learning Approaches, and Self-Efficacy Beliefs at the University of the South Pacific: A path analysis

approach. *Educational Psychology, 27*(6),789-806.

Ramalingam,V., Labelle,D., & Wiedenbeck,S. (2004). Self-Efficacy and Mental Models in Learning to Program. *SIGCSE conference on Innovation and technology in computer science education, (9)*,171-175.

Papadakis, S., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. (2014). Novice programming environments. Scratch & App Inventor: A first comparison . *2014 workshop on interaction design in educational environments*,1-7.

Schwartz, J., Stagner, J., & Morrison ,W. (2006). Kid's Programming Language. *ACM SIGGRAPH Educators program.*

Scherer, R. (2016). Learning from the past—The need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology, 7,* 1-4.

Van der Schaaf, M., Baartman, L., Prins, F., Oosterbaan, A., & Schaap, H. (2013). Feedback dialogues that stimulate students' reflective thinking. *Scandinavian Journal of Educational Research, 57*(3)*,* 227-245.

Yuen Lie Lim, & Lisa-Angelique. (2011). A Comparison of Students' Reflective Thinking across Different Years in a Problem-Based Learning Environment. *An International Journal of the Learning Sciences*, *39*(2)*,* 171-188.

Yünkül, E ., Durak, G ., Çankaya, S ., & Mısırlı, Z . (2017). Scratch Yazılımının Öğrencilerin Bilgisayarca Düşünme Becerilerine Etkisi . *Necatibey Eğitim Fakültesi Elektronik Fen ve Matematik Eğitimi Dergisi , 11* (2) , 502-517 .