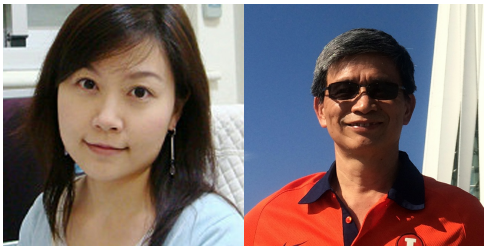
運算思維與中小學資訊科技課程

林育慈* 國立臺灣師範大學資訊教育研究所助理教授

吳正已國立臺灣師範大學副校長**



摘要

各先進國家為因應資訊科技的發展與國家人才需求，皆重新擬訂資訊科技課程之理念與方向。我國資訊科技課程歷經數次變革，十二年國教總綱中已將「資訊科技」列為國、高中之必修課程，課綱草案亦已將運算思維列為重要理念，希冀由運算思維之培養，提升學生善用運算思維與資訊科技工具解決問題、合作共創、溝通表達等高階能力。本文除探討我國及世界各國資訊科技課程發展理念外，並由程式設計與STEM理念分別提出運算思維教學的實施策略，藉以拋磚引玉，喚起各界對資訊科技教育之重視與投入。

關鍵詞：資訊科技課程、運算思維、程式設計教學

前言

資訊科技之發展一日千里，影響所及，從日常生活的資訊化，到國家經濟發展與科技前景皆與資訊科技密不可分。因此，各先進國家隨著資訊科技之更迭，皆不斷調整資訊科技教育之走向與內涵。美國國會於2015年通過的每位學生成功法案（Every Student Succeeds Act, http://www.ed.gov/essa）與白宮於2016年提出的Computer Science for All計劃（White House, 2016）中，不但將資訊科學視為與語文、寫作、科學、數學等為同等重要之學科，更於其中突顯程式設計在所有學科扮演的重要角色，此一重大潮流實不容忽視。我國適值十二年國教課綱修訂之際，除須檢視現有資訊科技課程理念與內涵，更應根據資訊科技發展趨勢與人類福祉之需求，重新訂定合乎時宜的課綱，並思考於教學現場實施之相關策略。各國為因應資訊時代中運算需求的普及，紛紛於最新制訂的資訊科技課程中表明「運算思維」（computational thinking）的重要性，並於課綱中融入相關理念（CSTA, 2011；DOEE, 2013；ACARA, 2013）。我國亦於十二年國教科技領域課綱草案中，將運算思維視為資訊科技課程主軸。為使各界明瞭資訊科技課綱修訂之依據與理念，本文將從我國資訊科技教育歷年來的發展及各國資訊科技教育的趨勢，說明並探討在十二年國教資訊科技課程的設計依據與主要理念，並提供教學方法的建議，以使教材編輯與教學者有所依循。

壹、我國資訊科技課程的發展

我國資訊科技教育經過幾次的變革，逐漸由操作技能導向的課程演變為高階能力導向之課程（吳正己，2010），主要的變革分為四個階段：

一、程式技能導向的資訊科技教育

臺灣中小學資訊科技教育是從民國73年公布的高中的選修課程「電子計算機簡介」開始，課程目標主要為了解電子計算機之功能、原理及應用，並培養學生程式設計的技能與運用計算機處理資料的能力。由於時值個人電腦發展之際，欲利用電腦進行工作，多須透過程式設計完成，因而此時期之課程教學以讓學生熟悉電子計算機的基本操作與程式設計技能為主，以因應當時產業與國家發展之需求。因此即使是程式設計的課程，也多以語法結構之教學為主，希冀培養學生撰寫程式以處理資料之技能。

二、軟硬體應用導向的資訊科技教育

民國84年國中開始有了必修的「電腦」課，以培養全民之基本的電腦基本素養為

主要目標（教育部中等教育司，1995），由於時值個人電腦普及化，以及視窗作業系統與應用軟體的蓬勃發展，課程設計以培養學生認識、操作與應用電腦軟硬體為主。然而，國中時期既已奠定電腦素養之基礎，高中的選修「電腦」課程則納入電腦科學導向之內容（吳正己、何榮桂，1998），簡介電腦工作原裡、作業系統、程式語言、演算法與資料結構等主題，此時已由過去偏重操作與應用技能之課程走向逐漸改變為科學導向之課程內涵，希望能透過學習電腦科學各重要原理與原則，培養應用電腦解決問題之能力。

三、問題解決與電腦科學導向的資訊科技教育

為配合民國93年全面實施的九年一貫課程，中小學課程再次做了修訂，但國小、中資訊科技並未納入正式課程，而以資訊教育重大議題處理，目的在使學生了解資訊科技與生活的關係，並能運用資訊科技工具有效解決問題（教育部，2008a）。高中則於95年頒布課程暫綱（教育部，2006），98年正式頒布課綱，將「電腦」更名為「資訊科技概論」，此課程為避免前次過於強調電腦科學內涵而流於知識面的學習，更加強調邏輯思維與問題解決能力的培養，將電腦科學知識的學習視為培養邏輯思維與問題解決能力之途徑（教育部，2008b）。

四、運算思維導向的資訊科技教育

儘管98課綱強調培養學生邏輯思維與問題解決等高階能力，但這些高階能力與資訊科技學科的相依性卻不十分明確，泛論性的描述使得社會大眾無法明確認知資訊科技教育之意義及其重要性，教師亦難有明確之教材設計指引。事實上，資訊時代所需之有效應用資訊科技的能力為一相當獨特之高階能力，因此於十二年國教資訊科技課綱重新訂定之際，研修小組根據國際資訊科技教育之趨勢，希望將這些高階能力能夠如同數學、科學等其他學科所培養的能力般明確定義，此即所謂的「運算思維」。此次課綱修訂便以培養學生運算思維為主要課程理念，希望學生能透過動手實作，有效利用運算思維與資訊科技工具解決問題、合作共創與溝通表達，此與總綱所訂定之核心素養中的系統思考與問題解決、規劃執行與創新應變、符號運用與溝通表達、及科技資訊與媒體素養等能力，皆密切呼應（國家教育研究院，2015）。

貳、各國資訊科技課程趨勢

隨著資訊科技的發展與教育理念的變遷，各國亦不斷修正資訊科技教育的目標與內涵。資訊科技發展之初，為求全民具備基本操作使用技能，資訊科技教育多以

培養國民資訊與通訊技術（Information and Communication Technology, ICT）相關技能為主要目標。但隨著資訊科技之普及，基本之操作應用能力已不敷生活與職涯之需求，因此美國資訊科學教師組織（Computer Science Teachers Association, CSTA）2003年所訂之K-12電腦課程標準（CSTA, 2003）、荷蘭2007年修訂之資訊科學課程標準（Grgurina & Tolboom, 2008）、以及德國電腦科學組織2008年所訂之中等學校資訊科學課程標準中（Brinda, Puhlmann, & Schulte, 2009），皆指出資訊科技教育不應僅止於資訊與通訊技術技能的培養，而應納入資訊科學內涵，以更全面理解與應用資訊科技。然而，面對二十一世紀的挑戰，國民已不能僅止於追求專精的知識或技能，更需要培養高層次的能力以因應複雜而瞬息萬變的未來社會。因此，美國二十一世紀關鍵能力聯盟訂定二十一世紀的關鍵能力包含：批判性思考與解決問題、溝通、合作共創、以及創造力（Partnership for 21st Century Skills-P21, 2007），資訊科技教育亦有此趨勢：美國國際科技教育應用協會（The International Society for Technology in Education, ISTE）所提出之國家資訊科技應用標準（National Educational Technology Standards, NETS）中亦指出，資訊科技應培養學生高層次的技能，包含：創造及創新、溝通及合作、研究及資訊運用（information fluency）、批判性思考、問題解決及決策、數位公民與科技操作及概念（ISTE, 2013）。然而，這些高階能力乃各學科領域皆應培養的共通能力，因此，為定義與表明資訊科技教育所能帶來的效益，許多學者引用Wing所提出的運算思維一詞（Wing, 2006），重新闡述資訊科學的內涵與資訊科技教育的意義。

美國CSTA於2011年重新修訂的課程標準中，將運算思維視為貫穿整個資訊科學課程的主軸。美國國家研究院（American National Research Council, 2011）與國際科技教育應用協會亦強調運算思維的重要性，認為有必要將運算思維的培養納入正式課程。英格蘭於2013年所訂定之運算（Computing）課程標準亦將運算思維定為課程的關鍵過程（key process），指出電腦科學的主要精神是運算思維（DOEE, 2013）。澳洲數位科技課程（Digital Technologies）中亦強調應培養學生在運算思維上的理解與技能，以發展應用數位科技以思考並實作問題解決之能力（ACARA, 2013）。College Board將於2016年實施之「AP資訊科學原理」（AP Computer Science Principles）課程中認為，應讓學生掌握資訊科學的中心思想、理解運算思維的概念並實作運算思維（College Board, 2014）。除了教育界對運算思維的重視，Google不僅對運算思維下了定義，亦開發一系列運算思維的教材（Google, 2015）。

除了逐漸重視運算思維之外，強調科學、科技、工程及數學跨科際整合學習

之STEM（Science, Technology, Engineering, and Mathematics）教育趨勢亦影響了資訊科技教育的課程與教學。2009年美國總統歐巴馬於「美國振興及再投資法案」（American Recovery and Reinvestment Act, ARRA）中強調STEM教育之重要性，指出STEM教育能增加高價值之產業與工作機會，並復甦國家經濟。美國於2012年更推展新科技教育計畫，預備在未來十年內培養100萬名科學、科技、工程和數學領域的學生。實施STEM教學時，可結合科學模擬、電腦實驗操作與科學視覺化等工具與科技，透過動手實作（hands-on）的歷程，整合STEM這四個領域進行跨科際學習，以提供學生解決實際情境中複雜問題的機會，進而培養二十一世紀之關鍵能力（Mataric, Koenig, & Feil-Seifer, 2007; Connor, Ferri, & Meehan, 2013）。在STEM的實作教學中，利用運算工具與程式設計讓學生進行探索與資料處理，能培養學生整合運算與STEM知識來解決問題的能力（Psycharis, 2013）。因此，「運算」（computing）位居STEM之核心角色（Chi & Jain, 2011; Henderson, Cortina, Hazzan, & Wing, 2007），英格蘭於2014年實施的運算課程中亦指出，運算具備理論基礎、數學結構與邏輯推理的應用（Mathematics），須利用科學方法量測與實驗（Science），須設計、創造、與測試作品（Engineering），且須了解、評鑑、與應用各種科技（Technology），因此同時跨越S、T、E、與M四領域，在STEM的教育熱潮中，運算思維能扮演科學、科技、工程、與數學之橋梁（DOEE, 2013）。

參、運算思維教學

儘管運算思維之重要性已在各國極力推展的各項政策與活動中不言而喻，大家對運算思維仍有許多疑義，尤其是如何將運算思維的培養落實於資訊科技課程之教學，因此，實有必要釐清運算思維的定義並提供教學方法的建議。

一、運算思維的定義

運算是「一種需要演算過程（algorithmic processes）、能從演算過程獲益或能產出演算過程，並有確切目標的活動」（ACM, 2005），包含：為了特定目的設計與建置電腦軟硬體系統；處理、結構化並管理各種資訊；利用電腦進行科學研究；讓電腦系統更具智慧；產生與使用通訊或媒體；搜尋與蒐集與特定目的相關之資料等。隨著資訊科技的蓬勃發展，運算不但影響了生活所需之食衣住行，更影響了國家經濟、產業等各個層面的發展。因此，在這個處處皆需運算的時代，如何善用運算工具改善生活並促進國家產業與經濟進步，是各國皆相當重視的議題。而透過運算思維，可掌握問題的可運算性（computability），亦即是否可利用運算工具解決，再進

一步有效利用運算方法與模式解決問題（Wing, 2006）。CSTA定義運算思維為「讓問題可利用電腦解決的一種方法」（CSTA, 2011）；澳洲資訊科技課程將運算思維定義為「利用數位科技設計與實作演算解法解決問題的思維」（ACARA, 2013）；英格蘭在運算課程中的定義則為「一種凌駕於電腦軟硬體之上，能針對系統與問題提出思考架構的思維模式」（DOEE, 2013）；Google則將其定義為「一種包含許多特性的問題解決過程，如：邏輯化的排序與分析資料、用循序的步驟產出解法，以及特定特質，如：能自信處理複雜度與開放式問題」（Google, 2015）。綜合各種不同的定義，筆者將運算思維定義為「能有效應用運算方法與工具解決問題之思維能力」。在資訊時代中，運算思維已成為分析與處理生活問題，以及探究各種領域知識之基本工具。

目前多數中小學生都能輕鬆使用各種電子與機械產品，教師應利用這樣的優勢教導學生使用運算工具，越早教導他們，越能提早提升運算思維（Wing, 2008），然而要教導學生那些運算思維、如何教導學生運算思維、以及如何設計相關課程，仍為有待討論的議題。自運算思維一詞被提出之後到2015年之間，許多學者提出運算思維應包含哪些元素，多數研究皆認為運算思維應包含抽象化（abstraction）、問題分解（problem decomposition）、模式化與模擬（modeling and simulation）及演算法思維（algorithmic thinking）（Wing, 2006; CSTA 2011; Barr & Stephenson, 2011; Grover & Pea, 2013; Selby & Woollard, 2014; Google, 2015）。運算思維雖是基於運算工具的一種思維模式，但已凌駕於工具之上，亦即，藉由資訊科學之學習可培養運算思維，而這樣的思維能力可應用於各領域；反之，各領域亦可利用一些教學設計，培養運算思維。除了資訊科技教學，許多學者或機構亦紛紛針對運算思維於各領域之教學模式提出建議，表1是Barr 與Stephenson （2011）為資訊科學、數學、科學、社會科學及語言藝術提供運算思維的應用範例，教材編纂者可根據教學主題發展適合的教材與教學活動供學生練習運算思維。另外，Computer Science Unplugged （Bell、Alexander、Freeman、Grimley, 2009） 強調不使用運算工具也可以學習資訊科學的活動，其目的是為彰顯「思維」之重要性。透過運算思維的學習，學生可在學習其他學科領域時，擁有更結構化、程序化、抽象化的思考模式，但必須強調的是，其他領域的學習或不使用運算工具的學習活動雖亦可輔助運算思維的培養，但畢竟完整的運算思維是架構於運算工具與理論的基礎上，搭配資訊科技理論與實務的學習，方能全面培養運算思維。

表1

運算思維元素於各領域之應用範例

運算思維元素	各領域應用範例				
	資訊科學	數學	科學	社會研究	語言藝術
抽象化	使用程序來封裝一組經常重複使用的指令；使用函數；使用條件敘述、迴圈、遞迴等	使用代數的變數；辨識應用問題中的基本事實；研究代數函數並與程式函數比較；使用迭代（iteration）來解決應用問題	為一個物理的實體建立模式	總結事實，從事實中演繹結論	使用明喻和隱喻；寫有分支的故事
問題解析	定義物件和方法；定義main和functions	利用表示式表達運算順序	對物種進行分類		撰寫大綱
資料表示	使用資料結構，例如：陣列（array）, 鏈結串列（linked list）, 堆疊（stack）, 佇列（queue）, 圖（graph）, 雜湊表（hash table）等	用長條圖、圓餅圖表示資料；使用集合、數列、圖等表示資料	從實驗資料做出結論	總結並表達趨勢	為不同句型呈現其樣式
	利用動畫呈現演				

模式化與模擬	算法，參數掃值（parameter sweeping）	繪製笛卡爾平面上的函數並修改變數的值	模擬太陽系運動	玩世紀帝國、Oregon trail	重現一個故事
演算法思維	學習經典演算法；針對某一領域的問題實作演算法	做長除法、因數分解；作加減法的進位	進行實驗程序		撰寫操作說明

資料來源：Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*, 48–54.

二、運算思維教學方法

十二年國教科技領域課綱草案所訂定之資訊科技學習表現包括四個向度：運算思維與問題解決、資訊科技與合作共創、資訊科技與溝通表達及資訊科技的使用態度。運算思維為課程主要理念，應貫穿所有學習表現，亦即，教學設計應留意運算思維的培養是否同時促進合作共創、溝通表達、資訊科技工具之應用等能力。為達此目的，筆者由程式設計與STEM二個角度提供運算思維教學方法之建議。

1、程式設計與運算思維

儘管運算思維並非等同於程式設計，但程式設計是創造運算作品的主要方式，亦是輔助運算思維中所需的認知任務的工具及展現運算思維能力的媒介（Grover & Pea, 2013）。因此，程式設計課程為實踐運算思維教學的重要途徑，透過撰寫程式，能實作運算思維中的抽象化、流程控制、模式化、遞迴、重覆、除錯等能力。許多研究開始探討如何利用程式設計教學培養問題解決與運算思維，課程設計著重在引導學生運用運算思維解決問題，而非程式設計細節（Howland, Good, & Nicholson, 2009；Lewis, 2010, Jonas & Sabin, 2015）。「低門檻，高極限」為選擇運算思維學習工具的重要原則，希望學生不需經過太多低階技巧的學習便能上手，而能將學習焦點放在高階思維。視覺化程式設計工具（如Scratch與Greenfoot）讓學習者容易地學會程式設計，且能專注於設計與創作，經歷創作、修改與使用的歷程，學生可體驗與應用運算思維，更進一步可接續一般的程式語言（如Python、Java、Scheme與C），以逐步發展運算思維（Grover & Pea, 2013）。此

外，教師亦可讓學生透過視覺化程式設計工具製作互動式作品（例如：情境故事、動畫、遊戲與模擬軟體），學生於專題製作過程中須使用運算思維解決情境中的問題並完成作品（Brennan & Resnick, 2012）。在CSTA的課程中，國高中階段的運算思維教學須讓學生能設計解題的演算法、定義解題指令、分析指令、描述資料的表示與儲存、解釋循序與重覆等結構、以模式化與模擬表達問題情境，而這些學習表現都能藉由程式設計教學逐步培養。透過程式設計的學習，能讓學生從運算方法的設計、創作與實作中了解資訊系統的組成與運算原理，並能解析並評估問題的解法，進一步培養利用運算工具解決問題的能力。因此，程式設計教學在培養學生運算思維的歷程中是不可或缺的重要方法。此外，十二年國教科技領域課綱草案之資訊科技學習內容除了程式設計，亦包含演算法、系統平臺、資料表示處理及分析等面向，在利用程式設計進行運算思維教學時，亦應與其他面向之主題配合，例如：利用資料處理與分析的演算法撰寫程式以分析生活周邊的資料，或利用開放式硬體之程式設計模擬系統平臺之運作機制。再者，為達到學習表現，教學設計亦應多以合作學習或專題方式進行，讓學生有機會可利用程式設計溝通與表達其思維，並與同儕共同解決問題、合作創造運算作品。

繼美國總統歐巴馬在2014年大動作地親自宣導「一小時程式」（Hour of Code, https://hourofcode.com/），新加坡總理李顯龍亦於2015年在Facebook上分享了自己撰寫的數獨解題程式，各先進國家皆因深知程式設計能力對國家競爭力的影響而紛紛大力倡導程式設計教學之重要性，我國實應及時跟上腳步，不僅在國家的教育政策上應有所作為，全民亦應正視其重要性，莫使此一攸關國家發展的重要工作隱沒於升學主義的桎梏之下。

2、STEM與運算思維

資訊科技本身就是一門STEM學科（DOEE, 2013），但透過其他STEM學科的結合，學生可有更多機會整合運用各種運算思維，並更能體會運算的能力與實用性。然而，由於多數教師缺乏STEM跨領域知識，融入STEM的運算思維教材的設計並非易事，而十二年國教資訊科技草案擬定的選修課程「機器人程式設計」則提供一個科際合作的範例，讓學生可透過程式設計，了解機器人控制、感測器資料存取、通訊單元資料傳輸等有關生活科技課程中機電整合的知識與技能。資訊科技教師可根據個人興趣與專長，設計與發展跨STEM學科的運算思維教材，以提供學生更生活化的學習情境，及更豐富多元的創作環境。值得注意的是，如同程式設計教學，為達資訊科技課綱草案所訂定之學習表現，融入STEM的運算思維教學亦不應僅止於知識

（如物理公式）與技能面（如物理模擬軟體操作）之學習，而應留意是否給予學生解決運算問題、溝通與表達運算概念與程序、及與同儕進行合作共創之機會。

除了在資訊科技教學中導入STEM， STEM課程中亦可導入運算。資訊科技發展至今，科學、工程等研究已無法與運算分離（Hambrusch, Hoffmann, Korb Haugan, & Hosking, 2009），例如：進行天文研究時，天文望遠鏡所取得的大數據需要進行分析，或是氣象模型或工程材料的大量模擬等，因此在進行科學與工程教學時，導入運算思維，可幫助學生有效利用運算工具並藉由各種運算思維：抽象化、系統化資訊處理、符號表示、演算法表達與流程控制、模組化、效能限制及除錯等，能結構化地解析並處理科學與工程領域的複雜問題。美國國家科學基金會（National Science Foundation, NSF）與西北大學執行了一個Computational Thinking in STEM （CT-STEM, http://ct-stem.northwestern.edu/）計畫，設計了一系列STEM的課程來教導高中生運算思維，同時也舉辦了許多教師工作坊，以進行CT-STEM教學之訓練，每個課程單元皆包含STEM四種領域的知識，以及運算思維元素（包括：資料分析、模式化與模擬、運算問題解決及系統化思考），希望學生能利用運算工具學會各STEM領域知識、引導學生分析STEM情境中的運算問題（可使用或不使用電腦）、教導學生分析實體世界複雜問題與資料並發展解題演算法、試探學生對CT-STEM職業與研究的興趣、以及了解STEM領域中的運算思維。美國加州大學戴維斯分校成立了C-STEM（Computing-STEM, http://c-stem.ucdavis.edu/）中心，進行許多運算融入STEM的課程開發與研究。Hambrusch、Hoffmann、Korb、Haugan與Hosking（2009）為科學科系的學生設計一套運算思維的課程，利用程式設計工具（Python的變數、字串、陣列、資料型態轉換、數學運算、利用VPython繪圖、函數、參數等）以及運算工具與方法（數值軟體、模擬方法、物理模擬軟體、生物資訊工具），並納入電腦科學議題（物件導向設計、科學史、可運算性、未來的運算模式、DNA運算等）來進行教學，其中包含：數位語音處理、進行網格滲透（percolation in grids）的運算實驗、模擬物理系統與分析蛋白質交互作用等專題。在STEM課程中導入運算，除了能整合運算思維，亦能提供學生實作的機會，許多數學（Hiebert & Lefevre, 1986; Long, 2005；Yilmaz & Yalçin, 2011）、科學（de Jong & Ferguson-Hessler, 1996；Yilmaz & Yalçin, 2012）、及工程（Mataric, Koenig, & Feil-Seifer, 2007）等學科的研究皆指出應利用實作活動培養學生概念應用與程序操作。我國中小學的科學、數學與科技課程亦可導入這樣的跨領域教學模式，讓學生善用運算處理與分析不同領域的資料、更深入理解各領域的知識、體驗各種領域知識與運算在解決實體世界複雜問題時扮演的角色、並培養對科學、科技、工程與數學之

興趣。

肆、結語

我國為因應資訊科技發展趨勢與教育潮流，歷經多次資訊科技課程理念與方向的調整與內容的修訂，此次十二年國教資訊科技課程以運算思維為主軸，希冀培養學生利用運算方法與工具有效解決生活與職涯中各式問題的能力。然而，因應最新發展趨勢而訂定之課綱，內容難免涵蓋新興詞彙，使各界對於課綱之新理念感到艱深晦澀（何榮桂，2015），因此本文針對課綱草案訂定之沿革、原則與理念作一說明，並提供課程實施相關建議供教科書撰寫者及現場教師參考。

新課綱的推展勢必造成教學現場的壓力，但這卻也為臺灣的資訊科技教育帶來新契機。過去因過份重視升學考試科目，造成中小學的資訊科技教育偏重應用軟體的教學，缺乏思維能力的培養，學生在進入職場或高等教育時，普遍欠缺問題解決、獨立思考、創造力等高階能力，資訊科技課程以培養運算思維為課程主軸，透過適合的教材與教法，可培養學生有效利用運算思維與工具解決問題之能力。然而新課程的實施將造成資訊科技教師在人力與專業發展上的需求，因此政府有關單位應立即研擬師資培育與教師專業發展等配套措施，以因應教學現場之需求。此外，此次總綱並未規劃國小科技領域課程，而美國、英格蘭與澳洲等先進國家皆於國小階段即納入資訊科技課程（CSTA, 2011；DOEE, 2013；ACARA, 2013），希望從國小開始培養學生運算思維，能了解演算法與程式如何讓運算工具正確地執行指令並解決問題、了解運算系統與平台的基本運作方式、了解資料的儲存與組織等概念、並建立負責且安全的資訊科技使用態度。十二年國教科技領域課綱草案亦提供國小課程之建議，各校除利用彈性學習課程授課之外，亦可融入其他學習領域，例如：社會領域可利用資訊應用軟體進行資料分析與視覺化，以幫助學生呈現並了解社會與人文現象；自然領域可讓學生利用科學模擬軟體進行科學現象的觀察與分析，使其更深入明瞭科學現象與原理，以及其與科技、工程與數學之關聯性。如此雙管齊下，透過彈性時數授課與其他學習領域之融入，以及早培養學生運算思維，並建立使用資訊科技的康健習慣，期使學生具備自主學習資訊科技的能力，方能在運算時代中具備足夠的生活與職涯技能。

參考文獻

何榮桂（2015）。試論十二年國民基本教育「資訊科技」課程綱要規劃草案。

科學教育月刊，**250**，48-64。

吳正己（2010）。臺灣中小學資訊科技教育的沿革與現況。**中國教育技術協會資訊技術教育專業委員會第六屆學術年會暨海峽兩岸信息技術教育研討會論文集**（pp. 7-11），7月 26-29 日。西安，中國。

吳正己、何榮桂（1998）。高級中學現行電腦課程的內涵與特色。**科學教育月刊**，**208**，26-32。

國家教育研究院（2015）。**十二年國教科技領域「資訊科技」科目課程綱要草案**。未出版。

教育部（2006）。**高級中學課程暫行綱要 -- 高級中學選修科目「資訊科技概論」課程綱要**。臺北：作者。

教育部（2008a）。**國民中小學九年一貫課程綱要重大議題－資訊教育**。臺北：作者。

教育部（2008b）。**普通高級中學必修科目「資訊科技概論」課程綱要**。臺北：作者。

教育部中等教育司（1995）。**國民中學課程標準**。臺北：作者。

American National Research Council (2011), *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. Retrieved from http://www.nap.edu/catalog/13170/report-of-a-workshop-on-the-pedagogical-aspects-of-computational-thinking

Australian Curriculum, Assessment, Reporting Authority (ACARA). (2013). *Draft Australian curriculum technologies*. Retrieved from http://consultation.australiancurriculum.edu.au/Static/docs/Technologies/Draft%20Australian%20Curriculum%20Technologies%20-%20February%202013.pdf

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*, 48–54.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20-29.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting*, *Vancouver*, *BC*, *Canada*, 1–25.

Brinda, T., Puhlmann, H., & Schulte, C. (2009). Bridging ICT and CS: educational standards for computer science in lower secondary education. *ACM SIGCSE Bulletin*, *41*(3), 288-292.

Chi, H., & Jain, H. (2011). Teaching Computing to STEM Students via Visualization Tools. *Procedia Computer Science*, *4*, 1937–1943.

College Board. (2014). *AP Computer Science Principles Curriculum Framework*. Retrieved from http://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-curriculum-framework.pdf

Computer Science Teachers Association (CSTA) (2003). *CSTA K–12 computer science standards. The ACM K-12 Education Task Force*. Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/K-12ModelCurr2ndEd.pdf

Connor, K., A., Ferri, B., & Meehan, K. (2013). Models of Mobile Hands-On STEM Education Models of Mobile Hands-On STEM Education. *120th ASEE Annual Conference & Exposition*.

CSTA (2011). *CSTA K–12 computer science standards*. The ACM K-12 Education Task Force. Retrieved from http://www.csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf

de Jong, T., & Ferguson-Hessler, M. (1996). Types and qualities of knowledge. *Education Psychologist*, *31*(2), 105-113

Department for Education in England (DOEE) (2013, September 11). *National curriculum in England: Computing programmes of study*. Retrieved from

https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

Google (2015). Exploring Computational Thinking. Retrieved from https://www.google.com/edu/resources/programs/exploring-computational-thinking/

Grgurina, N., & Tolboom, J. (2008). The First Decade of Informatics in Dutch High Schools. *Informatics in Education*, *7*(1), 55-74.

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. doi:10.3102/0013189X12463051

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *ACM SIGCSE Bulletin*, *41*(1), 183–187.

Henderson, P. B., Cortina, T. J., Hazzan, O., &Wing, J. M. (2007) Computational thinking. In Proceedings of the 38th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07), 195–196. New York, NY: ACM Press.

Hiebert, J., & Lefevre, P. (1986). Conceptual and Procedural Knowledge in Mathematics: An Introductory Analysis. In J. Hiebert (Ed.), *Conceptual and Procedural Knowledge: The Case of Mathematics* (pp. 1-27). Hillsdale, NJ: Erlbaum.

Howland, K., Good, J., & Nicholson, K. (2009, September). Language-based support for computational thinking. In *Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on* (pp. 147-150).

ISTE (2013). *ISTE's NETS for Students*. Retrieved from http://www.iste.org/docs/pdfs/nets-s-standards.pdf?sfvrsn=2

Jonas, M., & Sabin, M. (2015). Computational thinking in Greenfoot: AI game

strategies for CS1: conference workshop. *Journal of Computing Sciences in Colleges*, *30*(6), 8-10.

Lewis, M. (2010). *Problem Solving through Programming with Greenfoot*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.186.959&rep=rep1&type=pdf

Long, C. (2005). Maths concepts in teaching: procedural and conceptual knowledge. *Pythagoras*, *62*, 59–65.

Mataric, M., Koenig, N., & Feil-Seifer, D. (2007). Materials for Enabling Hands-On Robotics and STEM Education. *AAAI Spring Symposium on Robots and Robot* Venues: Resources for AI Education, Stanford, CA.

Partnership for 21st Century Skills (2007). *Framework for 21st Century Learning*. Retrieved from http://www.p21.org/storage/documents/docs/P21_framework_0116.pdf

Psycharis, S. (2013). Examining the effect of the computational models on learning performance, scientific reasoning, epistemic beliefs and argumentation: An implication for the STEM agenda. *Computers & Education*, *68*, 253–265.

Selby, C., &Woollard, J. (2014) Computational Thinking: The developing definitions. *In Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE 2014*. ACM.

The Association for Computing Machinery (ACM) (2005). Computing Curricula 2005: The Overview Report. *The Joint Task Force for Computing Curricula 2005*. Retrieved from http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf

White House (2016). *Computer Science for All*. Retrieved from https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions on the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Yılmaz, İ., & Yalçın N. (2011). Probability and possibility calculation statistics for data variables (VDOIHI); statistical methods for combined stage percentage calculation. *International Online Journal of Educational Sciences*, *3*(3), 957-979.

Yılmaz, İ., & Yalçın, N. (2012). The Relationship of Procedural and Declarative Knowledge of Science Teacher Candidates in Newton's Laws of Motion to Understanding. *American International Journal of Contemporary*, *2*(3), 50-56.

---

* 林育慈，國立臺灣師範大學資訊教育研究所助理教授

** 吳正已，國立臺灣師範大學副校長

電子郵件：chihwu@ntnu.edu.tw；linyt@ntnu.edu.tw

來稿日期：2016年3月2日；修訂日期：2016年3月25日；採用日期：2016年4月20日