第一章 緒 論

第一節 背景

電腦科技的發展讓人們很容易使用套裝軟體來完成工作，而不需如早期需要就各項工作設計特定的程式才可運用電腦來完成工作。中小學生的電腦課大都以學習各種功能的套裝軟體為主，以期使學生能應用電腦完成輔助學習及完成日常生活中的一些工作。然而透過套裝軟體學得的技能往往不具恆久性，一旦套裝軟體有新版出現時，相關技能常需重新學習一遍。學者（Hartmann, Nievergelt & ReichRert, 2001）指出電腦的學習不應只是學習如何操作，而應是要對電腦的運作有所瞭解，因而提倡所謂的 4`R，即除了閱讀（Reading）、書寫（wRiting）、算術（aRithmetic）等三個在工業時代即一直強調的基本核心能力外，還需加上在資訊時代所需具備的程式設計的能力（pRogramming），這裏所謂的「程式設計能力」，除了字面上的含義外，也包含了電腦科學的基礎知識，因為程式設計是學習電腦科學的最主要工具，也是藉以了解電腦科學領域的最重要媒介。當學生對電腦運作及基礎知識有所瞭解之後，對於套裝軟體的運作原理也會更加深入，故能免於因套裝軟體功能不斷更新而產生了無法跟上的情況。

我國高中電腦課程標準中也蘊含相同的精神，課程目的是期待高中學生對電腦科學有一個均衡、完整的觀念。課程中特別重視程式設計單元（程式語言、演算法與資料結構），其所佔的上課節數約為 26 至 30 節，佔課程總時數的 42%，是電腦課程中的核心單元。學習程式設計的主要目的，除了了解程式語言的基本結構及演算法，更重視如何運用程式設計解決問題（吳正己、何榮桂，1998），期待學生在學習程式設計過程中，知道程式設計是一種演算法的表現，並能撰寫

程式解決特定的問題。

在一個高中電腦課程的調查研究（陳宏煒，2003）中顯示，目前高中電腦教師都視程式設計為一重要單元，且大部分的教師也都教授程式設計，主要使用 Visual Basic 作為講授程式設計的程式語言。然而在教導程式設計的課程，大都以程式語言的語法為主，所使用的例題多以示範語法的用處，忽略了模組化觀念的傳達，問題解決的方式、邏輯推理思考能力的培養，學生對程式設計沒有一個整體觀念，面對實際問題自然不易運用所學的在解題中，造成學生學習程式設計的興趣、意願不高（吳正己、林凱胤，1997）。

教師的教學方式是造成這樣的境況的許多因素之一，另一個原因是所選擇用以教導程式設計的程式語言，以目前高中電腦課程所使用的程式語言 Visual BASIC 為例，Visual BASIC 不但具有 BASIC 語言原有的語法，亦含括事件驅動、圖形介面設計，甚至包括部分的物件導向設計的概念，在一種程式語言中，含括了好幾種的概念，對初學者而言，是不容易掌握其中的概念（Cathy, 1998）。這也是學者認為初學者使用一般的程式語言（如 Visual BASIC、C、JAVA 等），在學習程式設計課程中的初期階段常會遇到的下列的困難（Brusilovsky, Calabrese, Hvorecky & Kouchnirenko, 1997）：

（1） 一般的程式語言語法複雜，對學習者的認知負荷不小。

（2） 一般的程式語言通常不會提供視覺的方式以呈現程式執行的歷程，學習者所能觀察到的過程是一個「輸入-輸出」的模式。亦即僅提供程式一個輸入，程式處理完畢後輸出結果，學習者對程式執行的整個歷程一無所悉。

（3） 學習初期階段所能使用的程式範例大多以說明語法為主，不易引起學

生的學習興趣。但若要使用與學生的生活經驗相關的範例，這類的程式一般來說都是比較大，學生又需要對該程式語言有一定程度的瞭解，才能了解程式的意義。

學者（例如 Brusilovsky, Calabrese, Hvorecky & Kouchnirenko, 1997）認為「迷你程式語言」（Mini-language）可以使初學者更易學習程式設計。所謂的迷你程式語言其指令少、語法結構簡單，可以讓初學者在較短的時間學習並運用這程式語言去解題。Pattis（1980）在其著作「Karel the Robot」以類似 Pascal 語言的語法設計了可以在虛擬世界中控制機器人的迷你程式語言，藉由這些語言提供的指令撰寫程式操控機器人，初學者可以學習程式設計的基本觀念，如循序執行（Sequential execution）、程序抽象化（Procedural abstraction）、條件執行（Conditional Execution）、重覆執行（Repetition）等觀念。另一個運用「迷你程式語言」原則是 Papert 的 Logo，Logo 語言可以說是第一套運用迷你程式語言的原則（Brusilovsky, Calabrese, Hvorecky & Kouchnirenko,1997），Logo 語言主要雖是為了提昇問題解決能力、創造力而設計，但卻也是一個可用於介紹程式設計的好工具，它使初學者在操控烏龜繪圖的過程中學習程式設計的基本觀念。

為了讓學生的學習運用 Logo 語言更有效，Papert 嘗試運用 Logo 語言來控制連接在電腦的機器人。學生首先是在控制面板用按鈕的方式以操控機器人（圖 1-1），接著學習運用 Logo 的指令，如 FORWARD 50（向前走 50 點）、RIGHT 90（向右轉 90 度）作同樣或更多樣化的操控。延續了運用程式語言操控實體機器人的構想，Papert 的實驗團隊與 Lego 公司共同合作進行 LEGO/Logo 專案，研究開發 LEGO/Logo 可程式化方塊（LEGO/Logo Programmable Brick），亦即是 Lego MindStorms 系列產品的原型（Martins, 1994）。

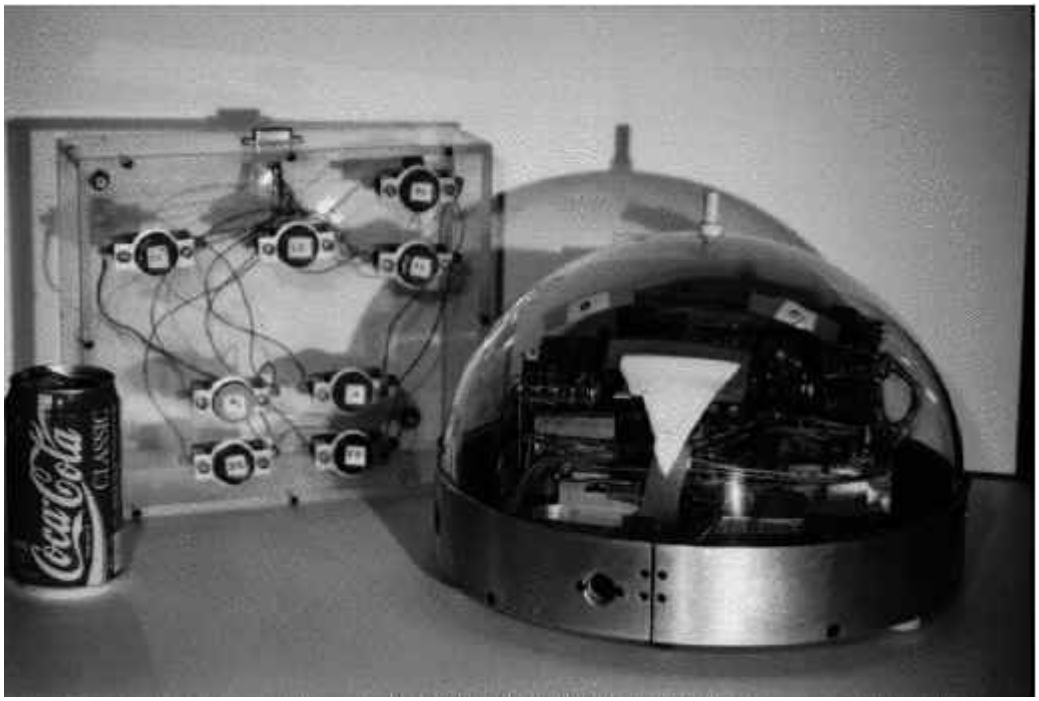

圖 1-1 連接在控制面板的機器車

（資料來源：Martin, 1994）

當 Lego Mindstorms 推出後，由於其組合方便，可以很容易組合各種形式的機器人，再加上可程式化的積木（Programming Brick）可以重新載入不同的韌體，從而創造出用不同的程式語言來控制 Lego 機器人，使得 Lego Mindstorms 的應用層面不再只侷限在兒童的玩具。Lego Mindstorms 也可以用作闡釋 ACM Computing Curriculum 2001 多個主題的合適平台（Klassner & Anderson, 2003），這些主題包括基礎程式設計課程、演算法、程式語言、計算機結構、作業系統、人工智慧和網路計算（Net-centric Computing）等。

許多學者（Canas, Bajo & Gonzalvo, 1994; du Boulay, O'Shea & Monk, 1989; Hoc & Nguyen-Xuan, 1990; Mayer, 1989; Mendelsohn, Green & Brna, 1990）認為程式設計初學者學習程式設計需要建立的心智模式--抽象的表徵機器模型（notional machine model）一個有效的機器模型應是簡單、能以相關的輔助工具來呈現程

式執行過程中的事件（Du Boulay, 1989）;能讓初學者運用程式語言來操控此輔助工具，以了解執行中程式的行為及其內裏隱藏的細節。故此有些學者（Barry, Laurence, & Thomas, 2001; Barnes, 2002; Fagin & Merkle, 2003; Lawhead, Duncan, Bland, Goldweber, Schep, Barnes & Hollingsworth, 2003）嚐試在程式設計課程使用機器人情境來闡述程式設計的基本觀念，以增進學生對程式的了解。

雖然有相關的文獻支持機器人在程式設計教學的應用，但到底其實際的教學成效如何卻少有實證的研究資料佐證，其中 Fagin 及 Merkle（2003）在大學的程式設計介紹課程，以實驗研究比較使用機器人與未使用機器人的班級。實驗組全學期使用 Ada 撰寫程式來控制 Lego Mindstorms，控制組則使用傳統的教學方式。研究結果顯示實驗組在學習成就上與控制組並無顯著差異。唯一的差別是實驗組學生覺得使用機器人來學習非常有趣、具挑戰性、與生活經驗相關。國內有關 Lego Mindstorms 的研究多是應用於問題解決能力方面的學習（如田耐青，1999；吳志緯，2003）。本研究嘗試運用 Lego Mindstorms 在高中程式設計教學，希望學生透過機器人的操控習得程式設計的基本觀念，進而探討其成效及 Lego Mindstorms 學習工具在程式設計教學中的使用操作相關事項。

第二節　研究目的

本研究探討運用 Lego Mindstorms 於高中程式設計教學的成效。主要目的有三：

（1） 探討運用 Lego Mindstorms 於程式設計教學對學生學習成就的影響。

（2） 探討運用 Lego Mindstorms 於程式設計教學對學生學習態度的影響。

（3） 探討運用 Lego Mindstorms 於程式設計教學的使用操作相關事項。

第三節　研究範圍

本研究中的實驗教學內容係以高中「電腦」科中的「結構化程式設計」單元為內容。