都會大學長者學苑

「耆樂簡單學程式」網上課程

第二堂：程式語言的發展歷史（教學短片）

視像文字稿

歡迎大家觀看「耆樂簡單學程式」網上課程的第二堂：程式語言發展歷史。今日我會簡介一下程式語言的發展歷史。大家可能已經有聽說過「電腦程式語言」及編程，平日也會在學習、工作和日常生活中接觸到。不講不知，我們現在看似輕易就能接觸到加入了「電腦程式」的產品，其實都是經歷過一段長時間發展出來的。現在就由我來向大家介紹一下「電腦程式語言」的「發展史」啦。

我們在這個單元的學習目標，包括電腦的演變、手動程式員的工作、機器語言及不同編程語言的種類（包括低階語言、高階語言和積木式編程語言）。以下時間，讓我們逐一為大家介紹。

首先是電腦的演變歷史。第一代電腦是指於 1946 年啟動，並以真空管為主要電子元件的電腦。當時電腦超過 1 萬 8 千個真空管，共重 30 噸重，是非常巨型的機器。處理資料的媒體為打孔卡片，而機器的指令種類及數量有限，所以只能完成少量的指令。

第二代電腦約於 1954 年出現，當時因為電晶體技術的發展，電腦的耗電量和散熱量得以有效優化，而電晶體的壽命和可靠性也比前一代的電子管要高。同時，在這個時期，程式編寫開始出現，科學家們開始製造出能夠處理日常工作事務的電腦，但其效率仍有待提升。

直到 1964 年，國際電腦商業機器公司（IBM）研發出集成電路的電腦，令電腦的體型變得較小，同時也能夠提升電腦的運算速度、記憶容量和事務處理的能力。然而，當時的電腦仍未普及到一般商業或工作用途。

由於電子工業的進步，1972 年成功研發了小型積體電路，一塊積體電路晶片上可以容納數千個甚至數萬個電子元件，這使得英特爾（Intel）的電腦以小巧的機身和低廉價格而快速進入各大公司、辦公室、商店、工廠或家庭。同時，很多小型的電腦亦逐漸出現，例如手提電腦、平板電腦、甚至智能手機等。

隨著電腦的誕生，程式員也應運而生。然而，在早期的電腦時代，電腦需要進行手動操作，要運用到編程來與電腦溝通，這項工作需要由程式員來進行。第一個編程語言是機器語言，它只包含兩個數字「1」和「0」。雖然後來我們開發了不同的編程語言，但我們仍然需要使用編譯器將它們轉換成「1」和「0」，以便電腦能夠理解和執行指令。

機器語言是由二進制代碼組合而成的，因為不需要翻譯過程，電腦可以直接執行指令，不需要加入中間的轉換過程，所以電腦處理速度較快。由於電腦唯一能夠理解的語言是由二進制「0101」等一系列由「1」和「0」組成的機器碼，例如，如果我們需要打印文件，就需要在電腦輸入一串長數字代碼，例如「10101」等等，這些代碼由指令和數據組成，電腦才能理解並執行「打印文件」的指令。

早期的程式設計者必須使用機器語言來編寫程式，他們需要將每個指令轉換成二進制代碼。這些複雜且難以理解的機器語言會導致程式的編寫非常困難，容易出錯，且修改和維護也很困難。因此在開發過程中的成本和時間都很高。為了克服這些問題，電腦科學家們開始嘗試開發其他編程語言，從而推動了程式語言的演變。

現在，讓我們來看看這些「程式語言演變」的過程吧！

常見的編程語言類別：電腦語言可以分為兩個類型，低階語言和高階語言。低階語言包括機器語言和組合語言，是比較接近機器的語言，指令集較少且較難理解。高階語言則相對較容易理解，語法接近人類自然語言，例如C++和 Java 等。

第一種出現的編程語言是機器語言，主要是二進制的指令集。例如，要打印文件，就需要以 1 和 0 的方式輸入指令，使電腦打印出文件。

第二種出現編程語言是組合語言，主要是由英文字母和數字組成的簡寫，指令集稍微複雜一些，但操作性仍然不高。

接著有高階語言出現，其語法已經接近人類自然語言，稱為程序式語言。例如，如果我們想要讓一個變數加上 1，只需要輸入"VAR+1"，就可以完成這個操作。這使得開發人員能夠更容易地理解和編寫代碼，並且可以處理更複雜的任務。

然後，為處理更複雜及大型的程式，有了物件導向程式語言的出現。物件導向程式語言主要將程式組織成相關的物件，每個物件都有其特定的屬性和行為，讓程式設計更模組化、可重用和易於維護，從而提高開發效率和讓程式看起來更加整潔易懂。

所以，為幫助初學者對編程更容易掌握，在 2000 年後，編程公司發明了「積木式編程」供初學者使用，它也是一種高階語言，但不需要像其他高階語言一樣輸入一連串的文字和數字。積木式語言是基於圖形化的積木組合介面，使用者可以通過拖動和組合不同的積木來編寫代碼。雖然背後仍

然需要輸入文字和數字，但這些操作由編譯器自動完成。因為電腦硬件如中央處理器只能夠讀懂機器語言，所有的高階語言都需要經過編譯器轉換為機器碼才能運行。

低階語言可以更有效地控制電腦的硬件，因此仍然被廣泛使用。現在還有很多驅動程式是使用低階語言編寫的。這些程式運算起來一般較為高效且佔用較少的記憶體。由於低階語言只有少量的指令，因此可以在沒有編譯器或直譯器的情況下直接轉換為機械碼。機器語言和組合語言是低階語言的主要例子。

高階語言能夠獨立於機器的程式編寫語言。使用人類能理解的文字，以近似人類表達的語言編寫。在程式開發的過程中，減少了處理複雜的機器結構，也減少了「翻譯」成機器明白的執行語言。開發人員可專注於設計，解決問題、達成想要的結果。

我們來使用同一個舉例吧 ，這個例子就是「保存數據」！當用戶想保存數據，只需要使用 CIN 的指令保存數據，開發完成後系統會編譯並轉成二進制。一般來說程式語言包括 C++及 Java 會使用編譯器，系統只需將程式編譯一次，以後便可以直接拿其代碼來使用。

積木編程語言是指將文本編程封裝成積木，學習者可以通過可視化界面輕鬆編輯和執行程式指令，拖動積木組合編程語法，在積木上設置參數功能。請看右上角的例子，我們只需要拖動對應積木，並說明在開始代碼後說 Hello 和往前 10 步，下面的物品就會按照程式預設流程一邊說 Hello，一邊往前 10 步。積木式編程一般使用直譯器，將高階語言的每一個述敘逐一翻譯成機器語言後直接執行。

經過剛剛的時間，我們已經說了目前常見的開發語言的優點和缺點，你們在後續的課程就會開始學習編制出自己的程式了。你認為哪一種編程語言最容易學習呢？

本影片由都會大學長者學苑提供內容及製作。