封面 Cover Page

教育部教學實踐研究計畫成果報告
Project Report for MOE Teaching Practice Research Program

計畫編號/Project Number：PGE1122172
學門專案分類/Division：通識學門
計畫年度：☑112 年度一年期 □111 年度多年期
執行期間/Funding Period：2023.08.01 – 2024.07.31

開發錯誤標示功能於線上自動程式批改系統以增進學習動機並配合程式除錯與可視化工具提升學習者除錯能力和幫助學習理解

C/C++程式設計

計畫主持人(Principal Investigator)：張傑帆

協同主持人(Co-Principal Investigator)：劉貞妤

執行機構及系所(Institution/Department/Program)：國立台灣大學/資訊工程學系

成果報告公開日期：□立即公開 ☑延後公開

繳交報告日期(Report Submission Date)：2024 年 9 月 20 日

本文與附件 Content & Appendix

開發錯誤標示功能於線上自動程式批改系統以增進學習動機並配合程式除錯與可視化工具提升學習者除錯能力和幫助學習理解

一、 本文 (Content)

1. 研究動機與目的 (Research Motive and Purpose)

1) 動機

隨著科技教育需求的增加，越來越多學生修習程式設計課程。為了提升大學生的跨領域程式設計能力，許多大學已將計算機程式設計課列為通識教育課程，特別是針對非資訊領域的學生，旨在提升台灣在數位經濟中的競爭力。這些課程逐漸成為跨領域通識教育的重要部分。在我教授的 C/C++程式設計課程中，修課人數持續增加。然而，批改作業與指出程式錯誤的工作十分耗時且勞力密集。為解決這一問題，我們引入了程式競賽常用的線上自動批改系統（Online Judge System），透過測資的通過與否，自動判斷學生程式的錯誤，減輕老師與助教的負擔。

雖然使用 Online Judge System 可以判斷出部分的錯誤類型，如 Compilation Error (CE)、Runtime Error (RE)、Time Limit Exceeded (TLE)和 Output Limit Exceeded (OLE)這些類型的錯誤以供使用者分析錯誤原因推測程式錯誤位置並進行 debug(Wasik et al., 2018)，但是針對 Wrong Answer (WA)這種類型的批改結果，因為部分的題目與練習題之中輸出的文本量較大或行數較多，造成部分學習者難以在這諸多的輸出當中找出自己程式錯誤的部分，或是因為粗心大意和針對文字觀察的敏銳度不足(Rawlinson, 2007)，無法察覺程式文本中的錯誤，如圖 1 所示，在留言板之中許多同學都表示自己撰寫的程式執行結果看起來與答案相同，然而卻無法通過批改，但實際上學習者所提交的程式的輸出與正解並不相同，然而學習者自己卻可能無法察覺，又或者在尋找錯誤的過程中，須逐一排查任何有可能造成錯誤的程式碼，在這眾多的輸出之中要花費大量的時間與耐心才能夠找到錯誤的地方，甚至可能誤會錯誤的部份把對的程式碼誤會成錯的，反而越改越錯，如圖 2 中的同學，想要找到程式中的錯誤卻歸因於錯誤的地方。這樣的情況增加了學生的學習挫敗感(Lawrance et al., 2010)，因此而降低了學習動機與成效，在另外一方面，學生找老師及助教問問題時，最常問的也是我的程式為什麼錯及哪裡有錯？在面對學生雜亂無章程式碼中，要先弄清楚是什麼樣的錯誤，尋找錯誤並試找出原因是一個十分艱難的過程。因此，本計畫將進一步開發自動比對功能，幫助學習者識別其程式輸出與正確答案之間的差異，讓錯誤更為顯而易見。我們希望將此工具整合至 Online Judge System 中，幫助學習者快速定位並修正錯誤，優化除錯流程，減少對其他程式碼的誤解。同時，這也能減少學生在尋找錯誤時的挫折感，提升學習動機，並提高自主學習能力，對老師和助教也能減輕負擔。

圖 1 同學們最常問的問題之一：學生認為看起來答案一樣，但其實並不相同，所以 Online Judge System 判定為 WA 錯誤，可是學生自己卻找不到。

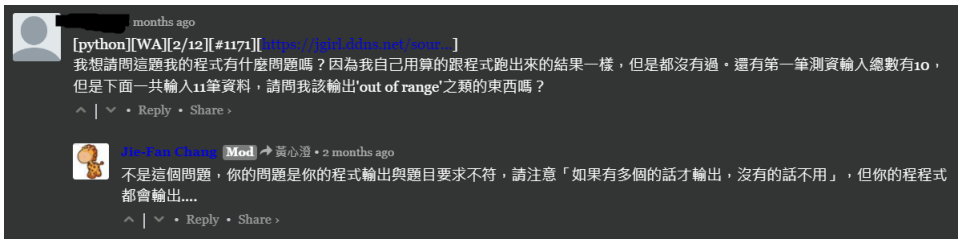

圖 2 同學們最常問的問題之二：學生誤判錯誤而找錯方向，其實是輸出不符合規範，但卻錯誤歸因於其他地方。

在程式除錯的過程中，找到錯誤後的下一步是釐清錯誤原因，這個過程稱為除錯（debug）。學習程式設計的過程中，學習者需要花費大量時間學習語法與設計邏輯，還需要耗費更多精力在練習和除錯上(Beller, Spruit et al. 2017)。debug 能力對程式設計學習者至關重要，良好的 debug 技巧不僅能幫助程式設計師快速解決問題，還能節省編碼時間，提升工作效率。因此，在程式教學中應該強調 debug 技巧的培養，讓學生學會如何解決程式中的問題，增強除錯能力。但傳統課堂通常以語法與範例教學為主，老師示範的程式碼經過設計，很少當場遇到 bug 開始除錯，學生在課堂上無法直接學到 debug 技巧(Cross, Hendrix et al. 2002)。學生通常只能在自主練習中摸索除錯方法，這使學習過程容易出現挫敗感。例如，學生常反映：「老師上課講的都聽得懂，看老師示範也很簡單，但自己寫的時候錯誤卻不停跳出，看起來和老師寫的幾乎一模一樣，卻總是找不到錯誤。」這樣的情況增加了學習挫敗感，進而降低學習動機與成效(Lawrance, Bogart et al. 2010)。debug 能力的培養需要大量經驗累積，這對初學者來說是一大挑戰(Jenkins 2002)，不少學生因此懷疑自己是否有天分而放棄學習(Gomes and Mendes 2007, Aissa, Al-Kalbani et al. 2020)。

2) 目的

為解決這個問題，本研究計畫在課程使用的 Online Judge System 中增加新功能。針對 WA 類型錯誤，系統將自動生成學習者程式碼中與正確答案不一致的輸出，提供行號提示並比對差異，並高亮顯示錯誤位置（如圖 3 所示）。讓學習者更容易發現錯誤，加快除錯流程，節省時間與精力，進一步提升學習動機。

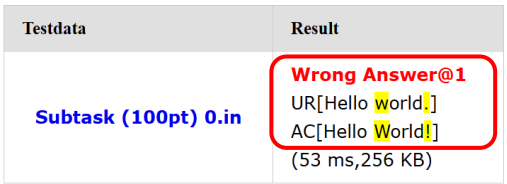

圖 3 系統 WA 錯誤提使功能示例。

Online Judge System 可以預先將程式預期的輸入與輸出預設先制定好成為測試資料集(測資)，並且將程式中要測試學習者所撰寫的程式是否正確的各種輸入進行測試，例如：請使用者撰寫一程式可以令使用者輸入身高、體重並計算出計算 BMI 值，並依其 BMI 值判斷出程度，學習者所撰寫好的程式必須在使用者輸入身高及體重後，可以輸出 2 行結果，第一行為依所輸入的身高、體重所計算出來的 BMI(輸出須四捨五入到小數點後 2 位)，第二行為依計算出來的 BMI 判定此 BMI 所代表的程度；因此在系統中可以先將各種 BMI 程度的輸入與輸出測資事先建立好(範例中有 6 種)，待學習者將其撰寫好的程式上傳進行測試，Online Judge System 將會對學習者上傳的程式進行 6 次測試，每一次執行將會輸入該次的輸入測資(in)(圖 4)，並檢驗學習者上傳的程式的輸出結果是否與該次的輸出測資(out)相同，以圖 4 為例，第一次測試時輸入身高 155 公分、體重 52 公斤，學習者所撰寫的程式就必須輸出 2 行文字，第一行為 BMI「21.64」，第二行為判定 BMI 等級「Normal」；第二次測試時輸入身高 180 公分、體重 99 公斤，學習者所撰寫的程式就必須輸出 2 行文字，第一行為 BIM「30.56」，第二行為 BMI 等級「Obese Class II」，依此類推直到測試完所有測資組，若學習者所撰寫的程式在這 6 組測試輸入 in 中都可以得到與預設的輸出 out 相同的答案，即可認為學習者所撰寫的程式通過了測試(Accepted)，系統會列出學習者所上傳的程式在哪些測資下無法通過，便可由此推斷出其所撰寫的程式錯誤的部分，以圖 5 為例，學習者在上傳其撰寫的程式之後被判定 Subtask 2 為 Wrong Answer，Subtask 2 的輸入為身高 175 公分 50 公斤，應輸出二行文字，第一行為 16.33，第二行為 Underweight，由此可以推斷出在 Underweight 的情況下程式有錯，提示學習者錯誤的輸出行號，圖中「@2」表示其輸出的第 2 行與答案不同，並高亮標示拼錯字的部份，第二行輸出的正確答案應為 Underweight (參照圖 4 中的 subtask 2 的 output)，學習者可以在其上傳的程式中，分析與判斷出輸出 Underweight 字樣的程式碼部份為程式的第 15 行，並檢查是否有錯誤，如圖 6 第 15 行程式中應輸出「Underweight」，但卻因其拼字錯為「Uderweight」造成錯誤，學習者可以藉由這些功能快速並精確的找到其撰寫的程式錯誤的部份，節省 debug 除錯所花費的時間與精神，不但可以幫助學習者找到錯誤，降低 debug 帶來的負擔，增進學習動機外，更可以有效的節省老師與助教的人力與精神損耗。

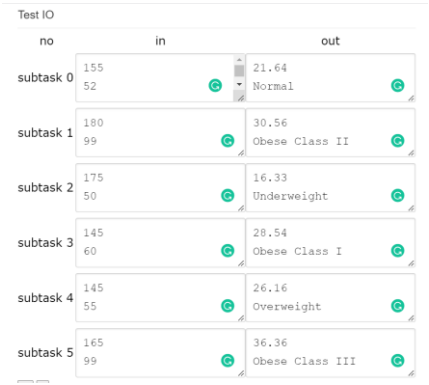

圖 4 測資範例，以 BMI 計算機為例，共有 6 組測資(subtask 0~5)，in 欄為輸入，out 欄為預設的正確答案。

Testdata	Result
Subtask (15pt) 0.in	Accepted(36 ms,128 KB)
Subtask (15pt) 1.in	Accepted(34 ms,128 KB)
Subtask (15pt) 2.in	Wrong Answer@2 UR[Uderweight] AC[Underweight] (26 ms,128 KB)
Subtask (15pt) 3.in	Accepted(18 ms,128 KB)
Subtask (20pt) 4.in	Accepted(18 ms,128 KB)
Subtask (20pt) 5.in	Accepted(30 ms,128 KB)

圖 5 測試結果範例，以 BMI 計算機為例，共有 6 組測資(subtask 0~5)，學習者上傳的程式無法通過第 3 組測資(subtask 2)，因為在 subtask 2 的輸入情況下，其所撰寫之程式的輸出第二行(@2)與答案不同，並高亮標示拼錯字的部份。

```c
#include<stdio.h>
#include<stdlib.h>

int main(){
    double h,w;
    double BMI;
    scanf("%lf", &h);
    scanf("%lf", &w);

    BMI =(double)(w/(h*h/10000));

    printf("%.2f\n", BMI);

    if(BMI < 18.5){
        printf("Uderweight\n");
    }
    else if(BMI >= 18.5 && BMI <24){
        printf("Normal\n");
    }
    else if(BMI >= 24 && BMI <27){
        printf("Overweight\n");
    }
    else if(BMI >= 27 && BMI <30){
        printf("Obese Class I\n");
    }
    else if(BMI >= 30 && BMI <35){
        printf("Obese Class II\n");
    }
    else if(BMI >= 35){
        printf("Obese Class III\n");
    }

    return 0;
}
```

<-第一行輸出為計算出來的 BMI 值

第二行輸出為 BMI 範圍代表意義(多選一)

圖 6 學習者所撰寫的部份錯誤程式，以 BMI 計算機為例，因第 15 行程式中 BMI 小於 18.5 時應於輸出第 2 行輸出 Underweight，卻誤植成 Uderweight (程式中第 15 行)而造成錯誤，錯誤行數提示功能可讓使用者在 34 行中快速的找到第 15 行有錯，然而在教學現場卻有許多學生因粗心或疲勞即便知道是第 15 行 Uderweight 這行有錯，卻也可能一時看不出錯在哪裡。(因為是範例，所以盡可能的挑簡單易懂的實際範例，實際教學場合的程式可能更加混亂與複雜，學習者更加不容易發現錯誤，還有計算類型的錯誤，但礙於篇幅就沒有列出)。

學習者在 Online Judge System 上進行練習的流程如下：

1. 閱讀與理解題目：看懂題目所規定的輸入與輸出需及測資限制。
2. 依題目要求設計演算法：學習者跟據題目所限定的時間與記憶體限制下設計能在有此限資源的條件下可以完成正確運算的程式。
3. 撰寫程式：依上述條件開始撰寫程式。
4. 本機測試：程式撰寫完成後，學習者應先使用範例測資進行測試。若出現 Compilation Error (CE)，表示語法不熟；若遇到 Runtime Error (RE) 或 Wrong Answer (WA)，則需要使用 debugger 進一步排查問題，進行除錯，如圖 7 紫色線條所示。
5. 上傳批改系統提交：當學習者在本機測試確認程式正確後，便可將程式上傳至 Online Judge System 進行最終驗證。若一切無誤，系統會通過測試（AC）；但若仍有錯誤，學習者會感到困惑與挫敗，尤其是在本機測試階段已多次確認無誤後，這時他們甚至可能懷疑題目或系統是否有問題。

Debug 除錯是學習程式設計中最困難且耗時的部分之一 (Jenkins 2002)，也是傳統教學中不易教授的技能(Lönnberg, Malmi et al. 2008, Wong and Jiang 2018)。本計畫旨在透過系統輔助工具，幫助學習者更有效地進行 debug，加速錯誤定位並找出原因。不同類型的錯誤需要不同的 debug 方法和工具，尤其是 Wrong Answer (WA) 類型的錯誤。當學習者認為程式已滿足題目要求並提交至 Online Judge System 進行批改，但系統卻判定輸出錯誤時，WA 錯誤提示功能可以幫助他們定位輸出與正確答案不同的部分，如圖 7 紅色線條所示。此功能能讓學習者快速找到問題，例如拼寫錯誤或理解偏差，並進一步修正程式，如圖 5 與圖 3 所示。

當學習者遇到更深層的程式問題，如語法或邏輯錯誤，或 Runtime Error (RE) 類型的錯誤時，debugger 工具可進行更詳細的內部錯誤排查。在「撰寫程式」與「本機測試」階段，若學習者無法定位 WA 錯誤或未進行充分測試導致 RE 錯誤，使用 debugger 可檢視程式的邏輯與變數變化，進行精確的除錯（如圖 7 紫色線條）。本研究除了開發 WA 錯誤標示功能，還希望透過引入 debugger 和改變教學方式，讓學生更容易掌握除錯技巧，降低學習的難度曲線，幫助他們度過學習程式的陣痛期，增進程式設計能力，並避免學習動機下降或中途放棄。

使用除錯工具能幫助初學者節省大量時間與精力，讓他們將更多精力投入語法學習和練習。因此，本課程將引入 gdb 進行流程錯誤檢查、valgrind 用於檢查記憶體洩漏，以及 VS Code C/C++ 擴充套件的除錯器教學。此外，我們還計畫導入程式設計「可視化」工具，幫助初學者更直觀地找到錯誤，增強學習信心。我們將進一步分析學生對不同除錯工具的偏好，研究他們在使用過程中的問題與習慣，並通過分析學生成績分佈，探討除錯工具與學習成效的關聯性。最終，本計畫旨在結合錯誤提示功能與 debugger，幫助學習者更精確地找到錯誤，減輕負擔，提升學習動機與除錯能力，並促進學生的程式設計理解與進步。

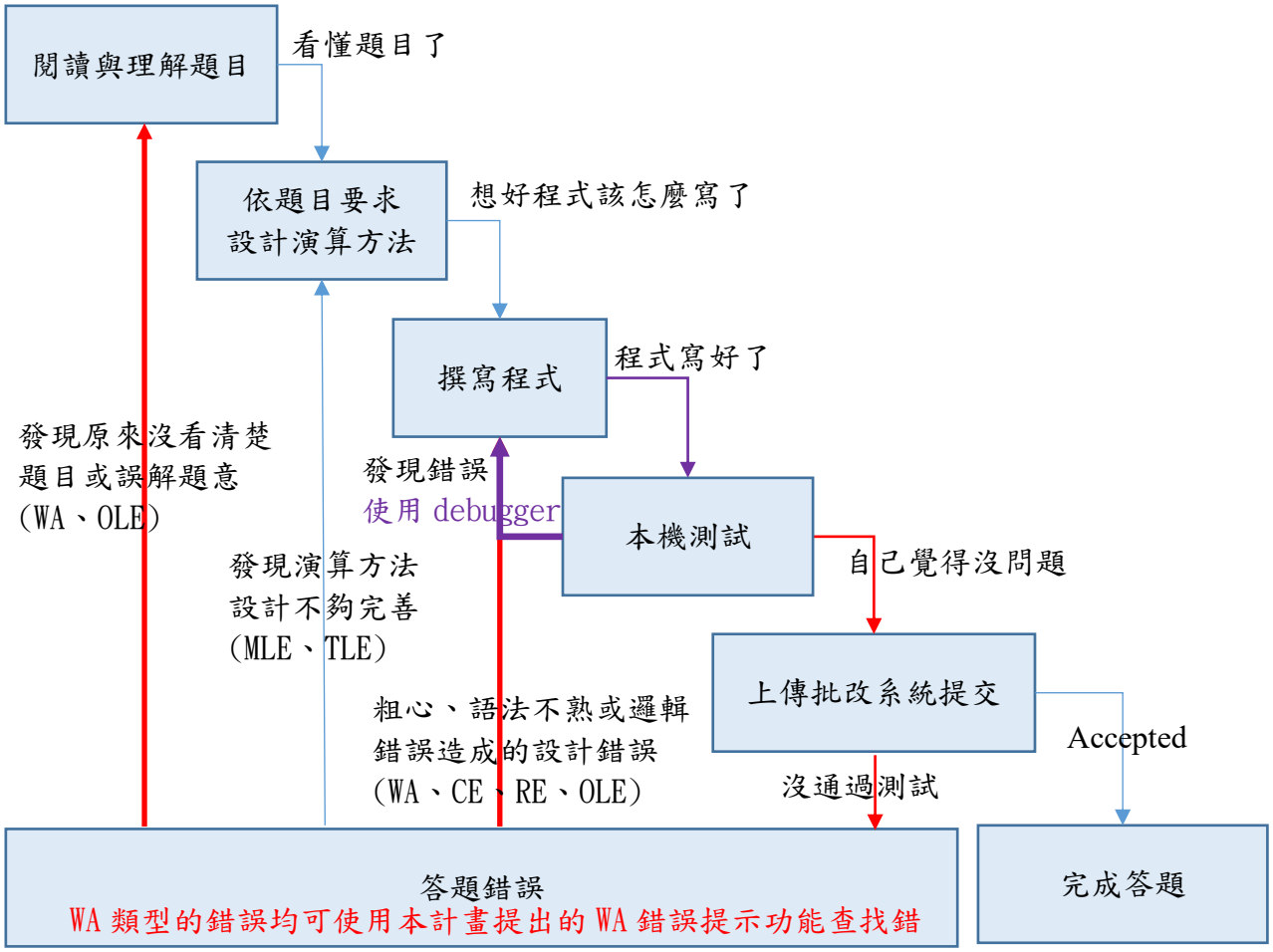

圖 7 學習者使用 Online Judge System 進行練習時的答題流程，在提交程式碼後對錯誤進行修正，直到答題完成(Accepted)為止。

2. 教學設計與規劃 (Teaching Planning)

教學目標

本課程旨在讓學生學會以下技能：熟悉基本的 C/C++ 程式設計邏輯與流程控制，能夠使用 C/C++ 進行檔案的讀取與寫入，並具備物件導向程式設計的能力。同時，學生將具備以下能力：能夠運用 C 語言進行計算並解決演算法相關問題，了解程式的時間與空間複雜度，並具備設計解決問題的演算法能力。

教學方法

本課程採用混成式授課，非同步線上部分使用 NTU Cool(NTU Course OnLine)平台，該平台提供影片教學與學習足跡紀錄等功能，每部影片時長為 5~20 分鐘不等，學生可依自身步調調整影片速度並提出問題。練習題將透過 Online Judge System 設計，學生可即時知道答題結果，錯誤時能利用 WA 錯誤提示功能進行除錯，提升學習動機與成就感。

實體課程中，教師可在教學後立即進行不計分練習，驗證教學效果，並根據學生學習狀況調整教學方式。教師可即時糾正錯誤，並以範例提醒其他學生避免相同錯誤。透過學習進度統計功能，學生可比較自身與同學的進展，提升成就感與學習動機，加強參與度與專注度，促進自主學習，教師也能掌握學習情況，針對性調整教學進度和內容，如圖 8。

圖 8 即時統計之學生答題情況，圓環為答題的狀況比例，綠色部份為 ACC 的提交次數，紅色為 WA，黃色部份為 CE。

本課程將使用台大資工系開發的開源 Online Judge System，Judge Girl 作為學生自主練習、課程教學、作業與考核的平台，評估學生程式除錯（debug）能力的進步情況。課程中將導入 gdb 進行流程錯誤檢查、valgrind 檢查記憶體洩漏，以及 VS Code C/C++ 擴充套件的除錯器教學，並進一步導入程式設計「可視化」工具，以提高學生的程式除錯能力。根據以往經驗，許多學生仍依賴 printf 進行除錯，而對 debugger 工具較為陌生。通過這些工具的教學，希望學生能有效使用 gdb 和 valgrind 取代傳統除錯方式，提升 debug 效率。可視化工具將幫助學生直觀了解記憶體和資料結構，快速發現並解決程式中的 bug(Guo 2013)，如圖 9。

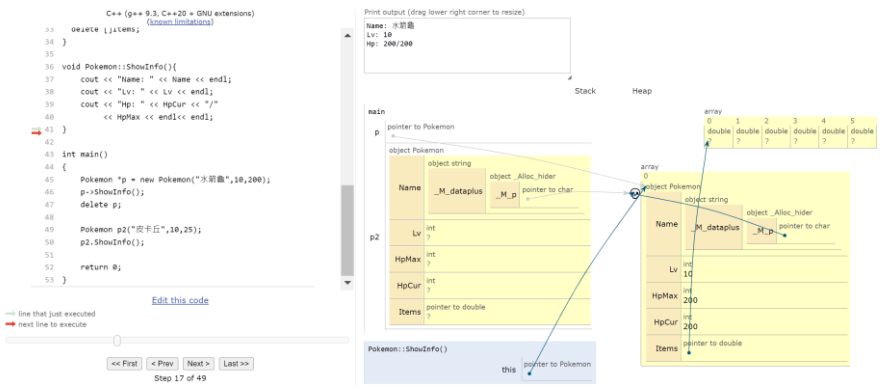

圖 9 導入程式運算及資料視覺化功能

學生成績考核與學習成效評量工具

該系統具備考試功能，稍加修改後亦可用於作業。教師可排定考試或作業時間，選擇試題並指定應試對象。系統會根據出題者設定的輸入資料，檢驗學生程式的輸出結果是否與預設答案相符，避免人工批改錯漏。此系統能全面測試學生程式在不同輸入情況下的完整性，更精確評估其程式設計能力，改善傳統批改因人力限制無法全面評量的問題。此外，系統提供即時統計表，教師可在考試過程中監控學生作答情況，並在考後根據數據反饋題目難易度是否合適。

3. 研究設計與執行方法 (Research Methodology)

研究步驟

本課程設計為線上與實體課程的混成模式，實體課程安排在單數周進行，雙

數周則為線上課程。實體課程除了補充教學內容外，將利用批改系統的統計功能，針對學生常犯錯的題目進行講解，並進行上機練習以驗證學生的學習效果。課程的核心程式設計語法和操作內容會預先錄製為線上影片，每段影片時長為 5 至 20 分鐘不等，學生可以靈活安排觀看時間。線上課程搭配自動批改平台 Online Judge System 增加互動性，學生在觀看完每段影片後可以即時進行練習並獲得即時反饋，若答錯，則可利用 WA 錯誤行號及內容標示功能進行除錯與修改，成功解題後繼續學習，形成良性學習循環，直到達成該周學習目標。

第一周的實體課程將進行課程簡介，說明課程進行方式、評量標準、影片觀看建議以及批改系統的使用方法與注意事項，並讓學生熟悉系統操作、帳號使用，以及如何利用系統提示、debug 工具和可視化工具進行除錯。在後續的實體課程中，老師會根據系統統計功能，找出學生在當周學習範圍內容易犯的錯誤，進行針對性講解，並安排上機練習活動，以檢驗學生的學習成效。課堂上，學生也可即時了解其他同學的學習進度，促進良性競爭，增強課堂氣氛和學習動機。雙數周的線上課程由學生自行規劃學習進度，需觀看指定影片並完成相應練習題與作業。課程中安排四次作業，每次約 20 題，學生需在作業公布後約 3 周內完成。作業題目將透過 Online Judge System 提交並自動批改。最後一周將進行期末考和問卷調查，考題當天公布，並透過 Online Judge System 進行線上考核。課程結束後，將對學生的考核成績、作業成績、平常練習以及問卷調查結果進行綜合分析，以評估課程的學習效果與教學成果。

課程成效評量工具

為測量學習者對本系統新開發的 WA 錯誤行號及內容標示功能是否對使用者的學習動機是否有所提升及 debug 及可視化工具實用程度，除錯能力是否因此提升和學習理解上的幫助，本研究將採用問卷評測學生自評其學習感受，為此本研究所採用之問卷參考(Polito and Temperini 2021)所使用的問卷，問卷內容見附件 1，並根據本課程與研究需求進行適度的修改，本問卷包含 38 個題目，題目各自對應到七個面向，以下簡述各題目對應到哪些面向:

(1) WA 錯誤行號及內容標示的機制使用體驗(experience)：13, 17 ,18 , 25
(2) WA 錯誤行號及內容標示的機制實用程度(usefulness)：4, 6, 7, 8 , 27 , 11, 12, 24
(3) WA 錯誤行號及內容標示功能學習者投入與動機程度(engagement)： 1, 2, 3, 5, 10, 22, 23
(4) Debug 及可視化工具實用程度，除錯能力是否因此提升(debugging ability)：28, 31,32
(5) Debug 及可視化工具對學習理解上的幫助(comprehensions)： 27, 29, 30, 34, 35, 36, 37
(6) 混成與自主學習體驗(blended learning and active learning): 14, 15 , 16, 33, 38
(7) 接受度(openness)： 19, 20, 21, 26, 31

成績考核

本研究將使用 Online Judge System 進行學生平常作業與期末考之成績考核工具，可以避免人工批改閱卷錯漏的情況，可以全方面的考慮各種輸入情況來測試學生所撰寫的程式的完整程度來給予評分，可以更全面完整的考核學生的程式設計能力，課程中的成績考核方式與配分為作業 (40%，回家作業)及期末考

(60%，考題由考核當天現場公布由線上批改系統進行線上考核)。

4. 教學暨研究成果 (Teaching and Research Outcomes)

(1) **教學過程與成果**

本次課程問卷結果

為探究學生使用本計畫開發的功能的學習動機是否提升，評估對批改系統的體驗以及評估學習者的學習動機及使用批改系統的感受。該問卷整體平均評分為4.05/5.0，即學生對批改系統給予正面反饋，如圖 10 所示。其中最高分為投入與動機程度(engagement) ：4.3/5.0，接著依序為 WA 標示使用體驗(experience) ：4.1/5.0、WA 標示實用程度(usefulness) ：4.1/5.0、Debug 能力是否因此提升(debugging ability) ：4.0/5.0、接受度(openness) 4.0/5.0，而最低分分別為 Debug 可視化工具對學習理解 comprehensions)：3.9/5.0 以及混成與自主學習(blended, active learning)：3.9/5.0。針對各面向於下方詳細說明之。

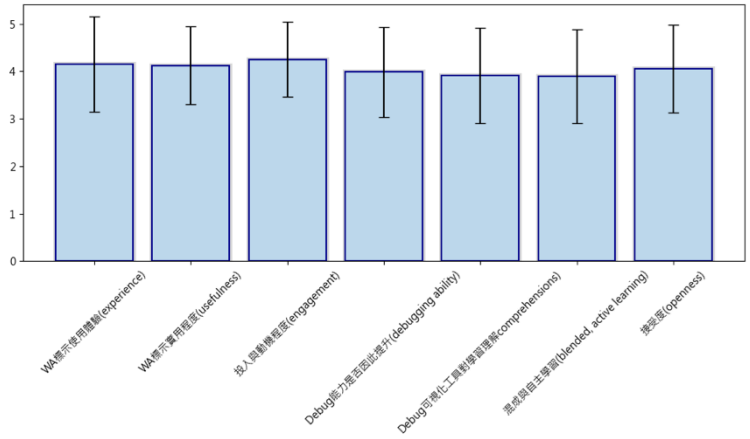

圖 10 問卷七面向分數分佈

A. 學習者投入與動機程度(engagement) 平均評分為 4.3/5(圖 11)。學習者表示他們知道有錯誤比對標示功能（4.3/5），因此他們常會去查看錯誤行號，了解哪裡出錯以及與答案的差異（4.3/5）。當通過批改系統的測試時會感到特別滿足(4.6/5)，而且有了錯誤標示功能後，會更加有動力去修正錯誤(4.1/5)，同時也因此想進行更多的程式設計訓練（4.0/5）。

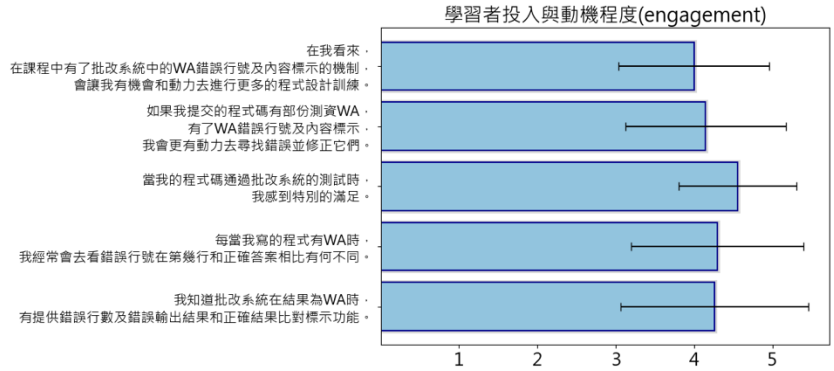

圖 11 學習者投入與動機程度(engagement)（最高為 5）

B. WA 錯誤標示的機制使用體驗(experience) 平均評分為 4.1/5（圖 12）。整體而言，學習者認為這是一個很好的經驗（4/5），而該機制具有創意（3.9/5），也非常實用（4.3/5），在解題與寫程式的過程中提供了很大的幫助（4.3/5）。

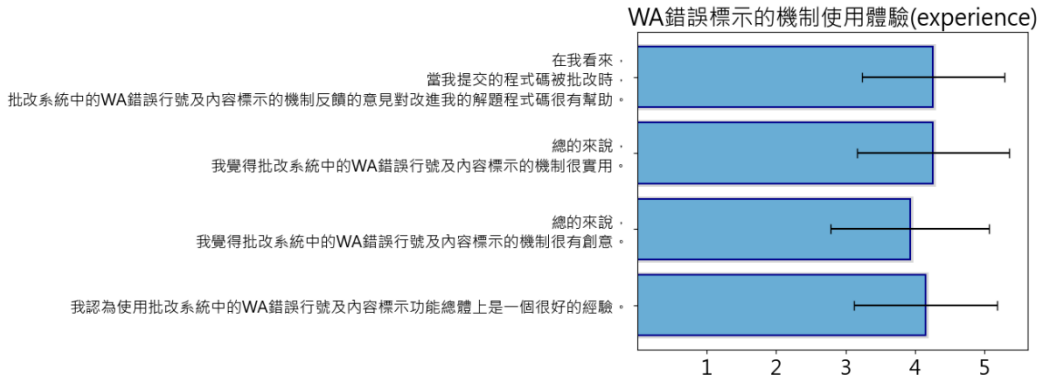

圖 12 WA 錯誤標示的機制使用體驗(experience)（最高為 5）

C. WA 錯誤標示的機制實用程度(usefulness)平均評分為 4.1/5（圖 13）。學習者認為 WA 提供即時回饋錯誤行號及內容的功能非常重要（4.4/5），且錯誤行號及內容標示相當容易使用（4.2/5）。這個標示幫助學習者更快找到程式中打錯字的地方（4.2/5）以及邏輯錯誤的部分（4.2/5）。此外，它還檢測出了在寫程式時未發現或未考慮到的錯誤（4.0/5），並有助於提升程式設計能力（4.2/5），寫出更好、更少 bug 的程式碼（4.2/5），對於提高程式設計能力非常有幫助（4.0/5）。

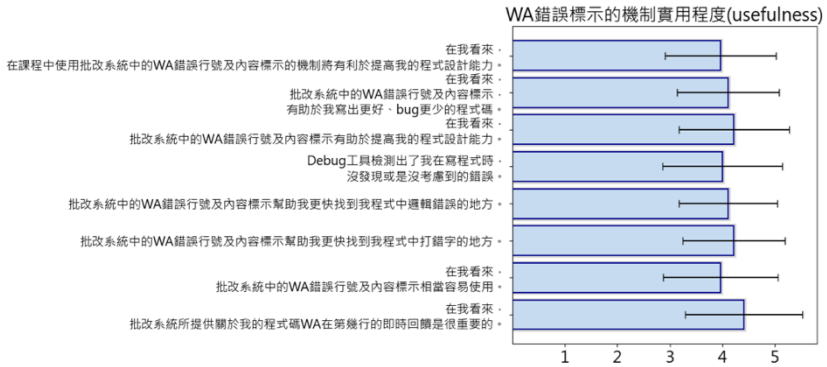

圖 13 WA 錯誤標示的機制實用程度(usefulness)（最高為 5）

D. Debug 及可視化工具提升除錯能力(debugging ability)平均評分為 4.0/5（圖 14）。學習者表示使用 debug 工具有助於提高程式設計能力（4.2/5），變得更擅長找到程式中的錯誤（3.7/5），同時也能協助寫出更好、bug 更少的程式（4.0/5）。

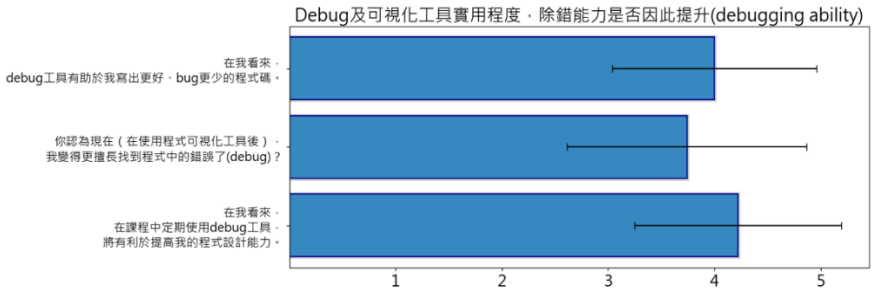

圖 14 Debug 及可視化工具提升除錯能力(debugging ability)（最高為 5）

E. 接受度(openness)平均得分為 4.0/5。如果沒有錯誤標示機制，學習者會感到更加疲倦（4.0/5），並且需要花費更多時間來解決問題（4.2/5），同時他們也希望其他課程中也能引入錯誤行號及內容標示功能（4.3/5）。因為有了錯誤標示機制後，學習者變得更擅長 debug（4.0/5），而且有了可視化工具後，debug 的能力也有所提升（3.7/5）。

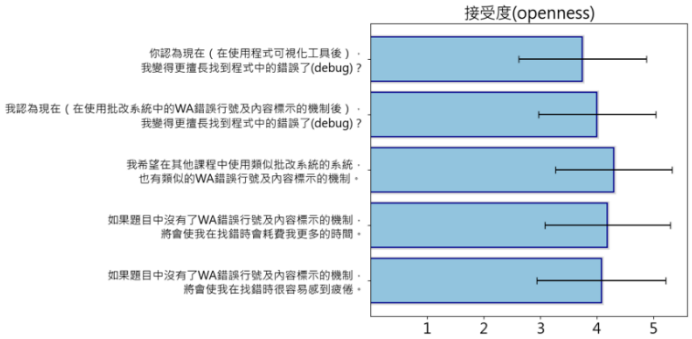

圖 15 接受度(openness)（最高為 5）

F. Debug 及可視化工具對學習理解上的幫助(comprehensions)平均得分為 3.9/5（圖 16）。使用 debug 工具能檢測出學習者未發現的錯誤（4.0/5），並且工具的反饋對改進程式碼非常有幫助（4.1/5）。debug 反饋可以更容易在網路上找到錯誤原因與解決方案（4.1/5）。可視化工具也讓學習者發現了未考慮到的錯誤（3.8/5），並且能更輕鬆找出程式中的錯誤（3.8/5）。同時，這些工具幫助理解程式的流程和原理（3.8/5），在教學中使用可視化工具也能更容易理解（3.8/5）。

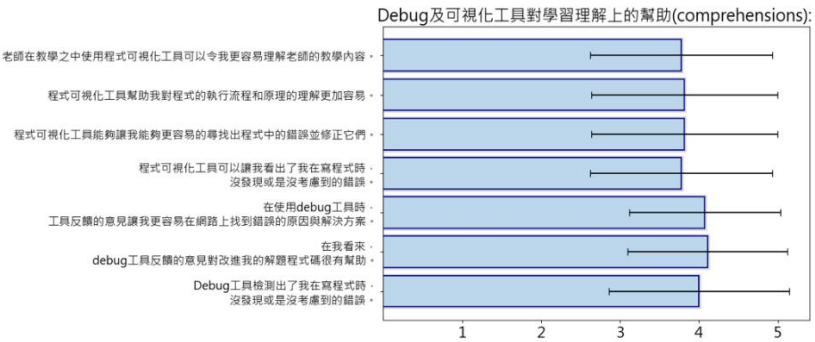

圖 16 Debug 及可視化工具對學習理解上的幫助(comprehensions)（最高為 5）

G. 混成與自主學習體驗平均得分為 3.9/5。這種學習方式讓學習者較少依賴助教、老師或同學幫忙找錯（4.0/5），並且在看完影片後進行練習時更加順利（4.0/5），同樣在完成作業時也變得更為順利（4.1/5）。然而，程式可視化工具的易用性得分為 3.3/5 相對而言還有改進的空間。

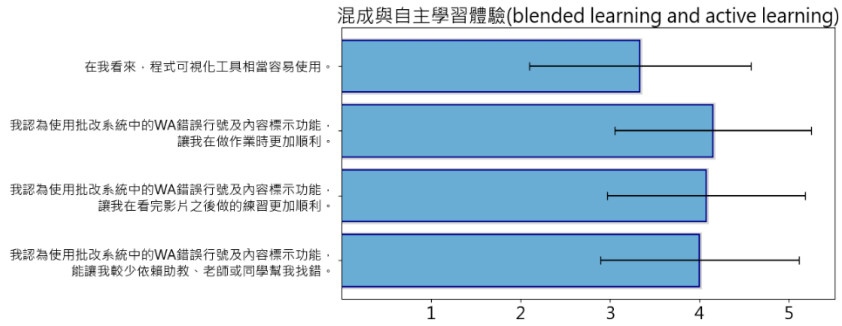

圖 17 混成與自主學習體驗各題得分（最高為 5）

問卷結果分析

每學期於課程的最後都會進行本系統使用的問卷填寫，除上述使用有 WA 錯誤標示 OJS 系統進行的班級外，與另一班級使用的批改系統無 WA 錯誤標示的班級進行問卷結果，在五個不同面向上的表現差異（圖 18、表 1），包括使用體驗、實用程度、投入與專注程度、混成與自主學習以及接受度後發現平均平分提升了 7.42%。結果顯示有 WA 錯誤標示的系統在每個方面的評分都比無 WA 標示的系統高，其中接受度的提升百分比最大，達到 15.97%，其次是「使用體驗」增加了 9.16%。這表明 WA 錯誤標示對於提升學習者的使用感受、實用性和學習投入具有顯著效果。

表 1 有無 WA 提示的 OJS 於五個不同方面的表現比較。

分析與比較	有 WA 提示	無 WA 提示	差異	提升%
使用體驗 (EXPERIENCE)	4.148	3.8	0.348	9.16%
實用程度 (USEFULNESS)	4.125	4.0	0.125	3.13%
投入與動機程度 (ENGAGEMENT)	4.252	4.1	0.152	3.71%
混成與自主學習 (BLENDED, ACTIVE LEARNING)	3.889	3.7	0.189	5.11%
接受度 (OPENNESS)	4.059	3.5	0.559	15.97%

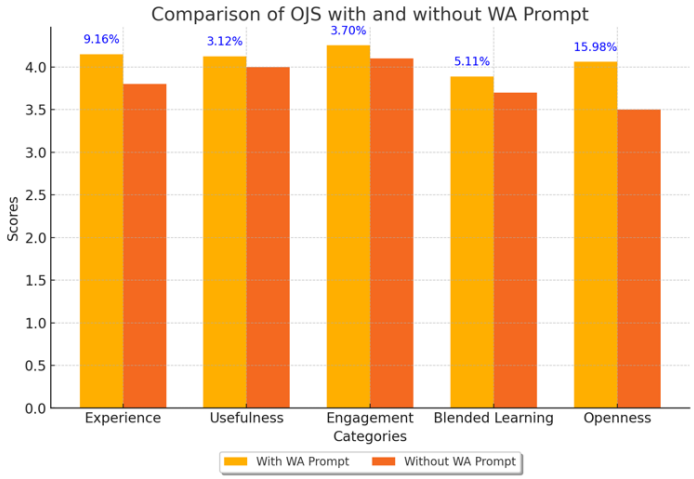

圖 18 有無 WA 提示的 OJS 於五個不同方面的表現比較

學習成效分析

對於學習成效的表現上，也對在有 WA 提示和無 WA 提示的 OJS 系統下的兩個班級的學生們，在四次作業（HW1 到 HW4）成績表現進行分析與比較（圖 19、表 2）。結果顯示，在有 WA 提示的系統下，學生在每個作業中的表現都優於無 WA 提示系統，進步百分比最大的是 HW4（13.57%），而 HW2 也有顯著提升（10.22%），期末考則因題目不同不列入比較。

表 2 有無 WA 提示的 OJS 於學生在不同作業和期末考試中的成績比較。

HW	有 WA 提示	無 WA 提示	差異	進步%
HW1	91.435	83.488	7.947	9.52
HW2	83.367	75.642	7.725	10.22
HW3	71.424	69.242	2.182	3.15
HW4	78.166	68.833	9.333	13.57

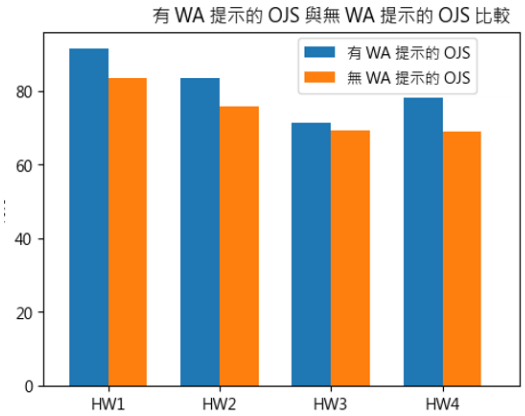

圖 19 有無 WA 提示的 OJS 於學生在不同作業的成績比較。

問卷最後也進一步調查學習者最常使用的 debug 工具（圖 20）。其中，48.1%的學生最常使用 print()進行除錯，14.8%的學生使用 gdb，而 40.7%的學生沒有使用任何 debug 工具，少數學習者（各佔 3.7%）使用 Copilot 或中斷點來進行 debug。

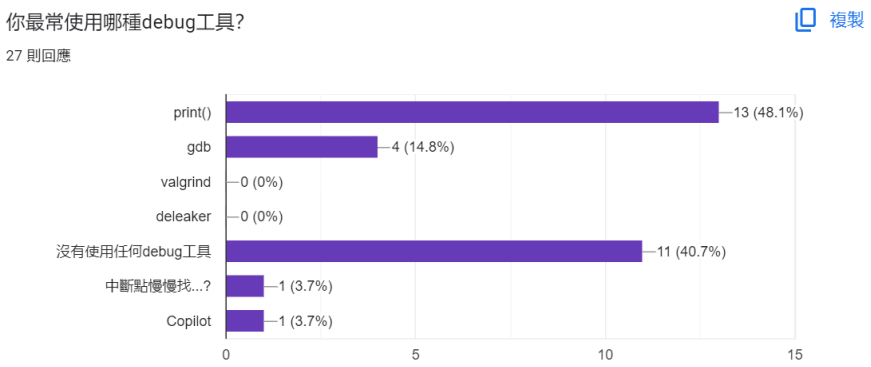

圖 20 最常使用的 debug 工具調查

同時也統計不同 debug 工具的使用人數、考試成績平均及解題數平均之間的關係（圖 21）。print()是使用人數最多的除錯方式，但不論使用何種除錯工具，在學習表現上的差異並不大，valgrind 與 deleaker 等專業的除錯工具，即便課程中有教學，但竟然無人使用，在訪問數名學生後得知，由於這兩項除錯工具使用門檻太高，不但安裝與使用非常繁瑣外更需要特殊作業系統才能安裝，因此在時間有限的修課時間下，學生們會選擇相對簡單易用的方式進行除錯，另外沒有使用任何除錯工具的學生學習表現是最好的，其原因並非是不用學習成效最好，而是其對課程中使用的程式語言的熟練程度已經很高，不再需要使用這些基礎的除錯工具。

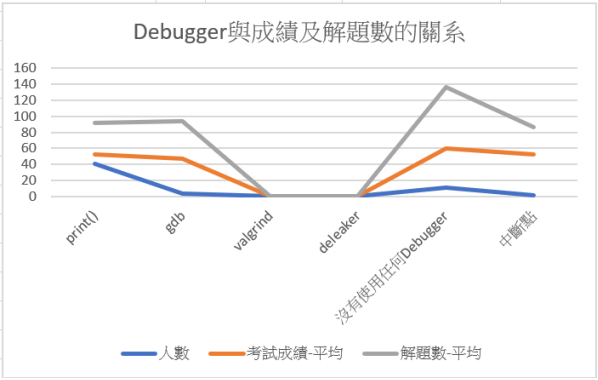

圖 21 debug 工具與考試成績及解題數量的關係

(2) **教師教學反思**

在混成課程中新增 WA 錯誤提示功能於自動程式批改系統對程式學習體驗和動機有顯著幫助。該系統提供了良好的輔助體驗，被學習者視為實用且有助於增進學習動機和自主學習能力。同時，使用批改系統進一步提升了教學效能和品質。新增 WA 錯誤提示功能於自動程式批改系統在混成課程中能有效提升學生的學習體驗、動機以及自主學習能力，進而促進整體教學品質的提升。

由本研究成果可得知，開發系統自動標示錯誤的功能性反饋給學生是十分重要且有用的功能，不但可以得到學習動機的提升外，還能帶來實質的學習成效的提升，但由於開發系統功能需要花費極高的人力與物力，所以一次計畫中無法完成所有的功能，希望於日後的計畫中能完成更多的功能，幫助學生們更好的學習。

(3) **學生學習回饋**

- 教學不用改善，該改善的是我這個看完影片還寫出一堆 bug 的 87。

- 混成教學提升學習效率，上課互動氣氛佳、幫助我能快速理解課程重點。
- 老師講話條理清楚、有邏輯，練習也能夠充分讓學生思考並應用上課所教，線上批改很好用能幫我快速的糾錯。很紮實的學習程式設計基礎。
- 老師很強，我有一些比較個人的問題都能知道可能的原因是甚麼並且給出建議，我能夠從建議中得到解決問題的方法。
- 系統功能實在太棒了，其他批改系統說我的程式錯我都不知道為什麼…
- 為什麼沒有 Runtime Error 的功能？常常 Runtime Error 都不知為什麼…
- 快被 RE 搞死了，我自己執行看起來都可以呀？到底為什麼 Runtime Error。
- 希望老師可以加入Runtime Error自動糾錯的功能，不然Segmentation Fault的因素太難找了。

5. 建議與省思 (Recommendations and Reflections)

本計畫開發的WA錯誤提示功能效果符合預期,對學生的幫助非常大,但錯誤的類型非常的多，不但有 Wrong Answer (WA)還有 Compilation Error (CE)、Runtime Error (RE)、Time Limit Exceeded (TLE)和 Output Limit Exceeded (OLE)等類型的錯誤，每一種錯誤的觸發原因與解決方式均不相同，需要耗廢學習者大量的時間與耐心，因此才造成了程式設計學習不易，並在自我效能感低下的情況下容易放棄的學習退縮情況。

本計畫開發的 WA 錯誤提示功能，成功的提高了系統的實用程度與接受度與學習者的學習動機、自主學習程度、學習成效等，但開發系統功能不易，仍有許多的難題等待攻克與對應的系統功能開發，如 Runtime Error (RE)之類的錯誤，希望可以將偵測此錯誤的功能實作在系統之中，並輔以大型語言模型給予學習者協助，以此可以給予學習者更多的學習上的支持，使學習者遇到錯誤時不再害怕與無助彷徨，能更有信心的繼續學習。

本系統為開源的程式,建議程式課程中都能使用,在計畫結案與論文投稿後,本計畫所新開發的功能也將會開放,讓所有教授程式設計課程的教師們都能使用。

結論

- **除錯工具**
  - 使用哪種對學習成效差異不大。
  - 初學者使用需求較高，熟練者使用的需求較低。
- **本研究開發了 WA 錯誤提示功能於自動程式批改系統之中。**
- **研究結果顯示：**
  - 該功能非常實用，具有極高的實用程度。
  - 該功能提供良好的使用體驗與接受度。
  - 有效提升學生的學習體驗、學習動機以及自主學習能力。
  - 有效提升除錯能力與學習成效。
  - 可以有效提升教學品質。

二、參考文獻 (References)

Bez, J. L., Tonin, N. A., & Rodegheri, P. R. (2014). URI Online Judge Academic: A tool for algorithms and programming classes. 2014 9th International Conference on Computer Science & Education,

Cross, J., Hendrix, T. D., & Barowski, L. A. (2002). Using the debugger as an integral part of teaching CS1. 32nd Annual Frontiers in Education,

Kurnia, A., Lim, A., & Cheang, B. (2001). Online judge. *Computers & Education*, *36*(4), 299-315.

Lawrance, J., Bogart, C., Burnett, M., Bellamy, R., Rector, K., & Fleming, S. D. (2010). How programmers debug, revisited: An information foraging theory perspective. *IEEE Transactions on Software Engineering*, *39*(2), 197-215.

Polito, G., & Temperini, M. (2021). A gamified web based system for computer programming learning. *Computers and Education: Artificial Intelligence*, *2*, 100029.

Rawlinson, G. (2007). The significance of letter position in word recognition. *IEEE Aerospace and Electronic Systems Magazine*, *22*(1), 26-27.

Trautman, E. (2015). Why Learning to Code is So Damn Hard. *Viking Code School Blog*.

Wang, G. P., Chen, S. Y., Yang, X., & Feng, R. (2016). OJPOT: online judge & practice oriented teaching idea in programming courses. *European Journal of Engineering Education*, *41*(3), 304-319.

Wasik, S., Antczak, M., Badura, J., Laskowski, A., & Sternal, T. (2018). A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)*, *51*(1), 1-34.

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

Wu, J., Chen, S., & Yang, R. (2012). Development and application of online judge system. 2012 international symposium on information technologies in medicine and education,

Zhao, W. X., Zhang, W., He, Y., Xie, X., & Wen, J.-R. (2018). Automatically learning topics and difficulty levels of problems in online judge systems. *ACM Transactions on Information Systems (TOIS)*, *36*(3), 1-33.

三、附件 (Appendix)

問卷設計

姓名：_____ 學號：_____ 班級：_____

5 = 非常同意, 4 = 同意, 3 = 沒有意見, 2 = 不同意, 1 = 非常不同意

1. 在這堂課之前我完全沒有學過程式設計。	5	4	3	2	1
2. 我知道批改系統在結果為 WA 時有提供錯誤行數及錯誤輸出結果和正確結果比對標示功能。	5	4	3	2	1
3. 每當我寫的程式有 WA 時我經常會去看錯誤行號在第幾行和正確答案相比有何不同。	5	4	3	2	1
4. 在我看來，批改系統所提供關於我的程式碼 WA 在第幾行的即時回饋是很重要的。	5	4	3	2	1
5. 當我的程式碼通過批改系統的測試時，我感到特別的滿足。	5	4	3	2	1
6. 在我看來，批改系統中的 WA 錯誤行號及內容標示相當容易使用。	5	4	3	2	1
7. 批改系統中的 WA 錯誤行號及內容標示幫助我更快找到我程式中打錯字的地方。	5	4	3	2	1
8. 批改系統中的 WA 錯誤行號及內容標示幫助我更快找到我程式中邏輯錯誤的地方。	5	4	3	2	1
9. 批改系統中的 WA 錯誤行號及內容標示測出了我在寫程式時，沒發現或是沒考慮到的錯誤。	5	4	3	2	1
10. 如果我提交的程式碼有部份測資 WA，有了 WA 錯誤行號及內容標示，我會更有動力去尋找錯誤並修正它們。	5	4	3	2	1
11. 在我看來，批改系統中的 WA 錯誤行號及內容標示有助於提高我的程式設計能力。	5	4	3	2	1
12. 在我看來，批改系統中的 WA 錯誤行號及內容標示，有助於我寫出更好、bug 更少的程式碼。	5	4	3	2	1
13. 我認為使用批改系統中的 WA 錯誤行號及內容標示功能總體上是一個很好的經驗。	5	4	3	2	1
14. 我認為使用批改系統中的 WA 錯誤行號及內容標示功能，能讓我較少依賴助教、老師或同學幫我找錯。	5	4	3	2	1
15. 我認為使用批改系統中的 WA 錯誤行號及內容標示功能，讓我在看完影片之後做的練習更加順利。	5	4	3	2	1
16. 我認為使用批改系統中的 WA 錯誤行號及內	5	4	3	2	1

容標示功能，讓我在做作業時更加順利。	
17. 總的來說，我覺得批改系統中的 WA 錯誤行號及內容標示的機制很有創意。	5 4 3 2 1
18. 總的來說，我覺得批改系統中的 WA 錯誤行號及內容標示的機制很實用。	5 4 3 2 1
19. 如果題目中沒有了 WA 錯誤行號及內容標示的機制，將會使我在找錯時很容易感到疲倦。	5 4 3 2 1
20. 如果題目中沒有了 WA 錯誤行號及內容標示的機制，將會使我在找錯時會耗費我更多的時間。	5 4 3 2 1
21. 我希望在其他課程中使用類似批改系統的系統也有類似的 WA 錯誤行號及內容標示的機制。	5 4 3 2 1
22. 在我看來，在課程中使用批改系統中的 WA 錯誤行號及內容標示的機制會大大增加我的工作量。	5 4 3 2 1
23. 在我看來，在課程中有了批改系統中的 WA 錯誤行號及內容標示的機制會讓我有機會和動力去進行更多的程式設計訓練。	5 4 3 2 1
24. 在我看來，在課程中使用批改系統中的 WA 錯誤行號及內容標示的機制將有利於提高我的程式設計能力。	5 4 3 2 1
25. 在我看來，當我提交的程式碼被批改時，批改系統中的 WA 錯誤行號及內容標示的機制反饋的意見對改進我的解題程式碼很有幫助。	5 4 3 2 1
26. 我認為現在（在使用批改系統中的 WA 錯誤行號及內容標示的機制後）我變得更擅長找到程式中的錯誤了(debug)？	5 4 3 2 1
27. Debug 工具檢測出了我在寫程式時，沒發現或是沒考慮到的錯誤。	5 4 3 2 1
28. 在我看來，在課程中定期使用 debug 工具將有利於提高我的程式設計能力。	5 4 3 2 1
29. 在我看來，debug 工具反饋的意見對改進我的解題程式碼很有幫助。	5 4 3 2 1
30. 在使用 debug 工具時，工具反饋的意見讓我更容易在網路上找到錯誤的原因與解決方案	5 4 3 2 1
31. 你認為現在（在使用 debug 工具後）我變得更擅長找到程式中的錯誤了(debug)？	5 4 3 2 1
32. 在我看來，debug 工具有助於我寫出更好、bug 更少的程式碼。	5 4 3 2 1
33. 在我看來，程式可視化工具相當容易使用。	5 4 3 2 1

34. 程式可視化工具可以讓我看出了我在寫程式時，沒發現或是沒考慮到的錯誤。	5 4 3 2 1
35. 程式可視化工具能夠讓我能夠更容易的尋找出程式中的錯誤並修正它們。	5 4 3 2 1
36. 程式可視化工具幫助我對程式的執行流程和原理的理解更加容易。	5 4 3 2 1
37. 老師在教學之中使用程式可視化工具可以令我更容易理解老師的教學內容。	5 4 3 2 1
38. 你最常使用哪種 debug 工具？(可複選)	□ printf() □ gdb □ valgrind □ deleaker □ 其他_____ □ 沒有使用任何 debug 工具