在线编程平台自动评测功能对 C 语言学习效率的提升研究

许镜雄

广东酒店管理职业技术学院 广东 东莞 523960

【摘　要】：针对传统 C 语言教学中反馈滞后、实践针对性不足、学生参与度低等核心痛点，本文结合建构主义学习理论、反馈学习理论与掌握学习理论，系统梳理在线编程平台自动评测功能的应用逻辑，从教学流程重构、任务体系设计、反馈机制优化、教师角色转型四大维度，提出 12 项具体实践措施。研究表明，通过“课前预习-课中互动-课后强化”的全流程措施嵌入，自动评测功能可有效解决传统教学痛点，帮助学生构建程序设计思维、提升编程实践能力，为高校 C 语言教学改革提供可操作的实施路径。

【关键词】：在线编程平台；自动评测功能；C 语言教学；学习效率；实践策略

DOI:10.12417/2705-1358.25.22.009

引言

C 语言作为计算机专业入门核心课程，是培养学生底层逻辑分析、程序设计思维的关键载体，但其语法严谨性、指针与内存管理的复杂性，使传统教学面临三重困境。随着“互联网+教育”的深化，在线编程平台（如 PTA、HDUOJ）的自动评测功能逐渐成为破解上述困境的关键工具。该功能可实时编译代码、通过多维度测试用例验证程序正确性，即时返回错误类型（如语法错误、逻辑漏洞、运行超时）与定位提示，支持学生反复提交修改，直至通过所有测试用例。但当前多数教学实践中，自动评测功能仅作为“作业批改工具”单一使用，未形成全流程、体系化的应用模式，其提升学习效率的潜力未充分释放。

1 在线编程平台自动评测功能对 C 语言学习效率的提升作用

1.1 实时反馈与精准纠错，加速知识内化

在 C 语言学习旅程中，错误是无可避免的“拦路虎”。传统教学模式下，学生提交代码后，往往需苦等 2-3 天才能获得教师反馈。在这段时间里，错误记忆逐渐模糊，错误认知也可能悄然固化，导致纠错困难重重。自动评测功能则如同一位不知疲倦的“即时导师”，打破了这种反馈滞后的僵局。自动评测功能则将反馈精细化，采用“错误类型+原因解析+修改建议”的模式。若学生出现数组越界错误，助力基础薄弱学生的错误修正成功率提升 50%以上，有效减轻因反复试错产生的挫败感，让学生在纠错过程中真正掌握知识要点，提升学习信心。

1.2 分层实践与个性化学习，满足多元需求

C 语言学习者群体基础参差不齐，犹如高矮不一的树苗，对知识养分的需求各异。基础薄弱者亟需扎实语法根基，基础扎实者渴望挑战逻辑难题，然而传统统一化作业如同“一刀切”，无法契合不同学生的成长需求，导致“优生吃不饱，差生跟不上”的尴尬局面。自动评测功能则借助“分层实践机制”，为每棵“树苗”量身定制成长方案。

1.3 高频实践与深度反思，锤炼编程思维

C 语言学习的核心目标，是培养学生用代码解决实际问题的实战能力。但传统教学常出现理论与实践脱节的现象，学生虽懂理论，却难以将其转化为有效的代码。自动评测功能通过构建“反复实践-即时反思”的学习闭环，引导学生从被动知识接收者转变为主动探索者，逐步锤炼编程思维。

2 相关理论支撑

2.1 建构主义学习理论

皮亚杰提出的建构主义理论强调，学习是学生主动建构知识的过程，而非被动接收信息。自动评测功能通过“提交-评测-纠错-再提交”的循环，让学生在反复调试中自主发现语法错误、逻辑漏洞，逐步构建 C 语言语法规则与程序设计思维，符合“主动建构”的核心要求，为“课前预习-课中互动-课后强化”措施设计提供理论依据。

2.2 反馈学习理论

布鲁姆的反馈学习理论指出，即时、具体、针对性的反馈是提升学习效果的关键：反馈越即时，学生对错误的记忆越清晰；反馈越具体（如错误行号、修改建议），学生越易定位问题根源。这一理论为“反馈机制优化”措施提供指导，要求自动评测的反馈信息需从“单一结果”向“错误解析+改进建议”升级。

2.3 掌握学习理论

卡罗尔的掌握学习理论认为，只要提供足够的学习时间与适当的帮助，绝大多数学生能掌握所学知识。该理论支撑“分

层任务设计”“个性化练习推荐”等措施，要求自动评测平台为不同基础学生提供差异化任务与资源，确保每个学生都能在“最近发展区”内逐步提升。

3 在线编程平台自动评测功能对 C 语言学习效率的提升措施

3.1 教学流程重构

3.1.1 课前预习

课前 3 天在平台发布“基础语法预习任务”，如“变量定义与数据类型匹配”“简单输入输出函数使用”，任务包含 5-8 道基础题，每道题设置 2-3 个测试用例（覆盖正常输入与常见错误输入）；要求学生完成任务后，截图保存自动评测的错误记录，标注“未解决的错误类型”（如“格式控制符使用错误”），带着问题参与课堂，实现“靶向学习”；教师课前导出平台数据，统计高频错误（如“忘记包含<stdio.h>头文件”），课堂重点讲解，提升教学针对性。

3.1.2 课中互动

讲解完核心知识点（如循环结构、函数调用）后，预留 15-20 分钟“即时练习时间”，在平台发布 2-3 道随堂题（难度低于课后作业），学生实时提交，教师通过平台大屏查看全班答题进度与错误分布；针对全班 50%以上学生出错的题目（如“for 循环边界值处理错误”），教师现场演示代码调试过程，结合自动评测的错误提示，解析错误原因；将学生分为 4-5 人小组，针对“个性化错误”（如“函数参数传递类型不匹配”），小组内讨论解决方案，修改后再次提交，教师巡回指导，培养协作调试能力。

3.1.3 课后强化

根据课前预习数据与课堂表现，将课后作业分为三级。覆盖课堂核心知识点，测试用例侧重“语法正确性”（如“用循环计算 1 到 100 的和”）；强化逻辑应用，测试用例包含边界值与异常输入（如“计算 1 到 n 的素数，n 由用户输入，需处理 n≤1 的异常情况”）；结合实际场景，测试用例侧重“代码效率与健壮性”（如“用指针实现数组去重，要求时间复杂度低于 $O(n^2)$”）；平台自动记录学生错题，生成“个人错题集”，每周推送 1 次“错题重练任务”，要求学生重新提交修改后的代码，通过自动评测验证掌握程度；每完成 1 个教学模块（如“数组”“指针”），在平台发布“模块自测题”，学生自主完成，平台即时生成成绩报告，标注未掌握知识点（如“指针与数组的关联使用”），引导针对性复习。

3.2 任务体系设计

3.2.1 情境化任务设计

将抽象知识点转化为生活问题，如讲解“分支结构”时，设计“超市折扣计算”任务（满 100 减 20，满 200 减 50，输入消费金额输出实付金额）；讲解“文件操作”时，设计“学生成绩录入与查询”任务（从文件读取成绩，计算平均分后写入新文件）；结合后续课程需求，设计“底层逻辑任务”，如讲解“内存分配”时，设计“简单链表创建与遍历”任务；讲解“函数指针”时，设计“简单计算器实现”任务（通过函数指针调用加法、减法函数）；同一知识点的任务按“基础操作→逻辑拓展→综合应用”递进，如“循环结构”任务：基础级（计算累加和）→提高级（打印菱形图案）→挑战级（解决“鸡兔同笼”问题）。

3.2.2 错误导向任务设计

针对 C 语言高频易错点（如“指针空值判断”“数组越界”“函数返回局部变量地址”），设计专项任务，每个任务包含 3-4 个“陷阱测试用例”，如“指针应用”专项任务中，设置“未初始化指针调用”“指针越界访问”等测试用例，引导学生主动发现错误；要求学生完成专项任务后，撰写“错误分析报告”，记录“错误类型、错误原因、修改思路、同类错误预防方法”，教师通过平台查看报告，针对性点评；每月组织 1 次“错题分享会”，学生通过平台上传典型错题与分析报告，全班交流，实现“一人错、众人学”。

3.3 反馈机制优化

3.3.1 错误提示精细化

将传统“答案错误”“运行错误”细化为具体类型，如“逻辑错误（循环边界缺失）”“语法错误（分号遗漏）”“性能错误（运行超时，建议优化循环）”“内存错误（内存泄漏，未释放 malloc 分配空间）”；除标注错误行号外，补充“错误上下文提示”，如“第 15 行：printf 函数格式控制符与参数类型不匹配（%d 对应 int，实际为 float）”；针对常见错误提供具体修改方向，如“数组越界错误：建议检查循环条件（当前 i<10，数组长度为 10，i 最大应为 9）”“函数未声明错误：建议在 main 函数前声明该函数，或包含对应头文件”。

3.3.2 多维反馈补充

在“正确性评测”基础上，增加“代码规范性评分”（满分 10 分），从变量命名（如“变量名是否见名知意，避免 a、b 等模糊命名”）、注释完整性（如“函数是否有功能说明，关键代码是否有注释”）、代码缩进（如“是否采用 4 空格缩进，代码块结构是否清晰”）三个维度评分，标注不规范之处；对挑战级任务，增加“时间复杂度”“内存占用”反馈，如“代

码时间复杂度为 O(n^2)，建议优化为 O(n)（可通过哈希表实现）”“内存占用过高，建议复用变量，减少不必要的数组定义”；针对优秀代码（正确性 100%、规范性≥9 分、效率最优），推送相关拓展知识，如“你的代码实现了链表反转，推荐学习‘双向链表反转’‘递归实现链表反转’等拓展内容”。

3.4 教师角色转型

3.4.1 任务设计者精准匹配教学目标

根据课程大纲，明确每个教学模块的核心目标（如“掌握函数参数传递”“理解指针与内存的关系”），设计与之匹配的自动评测任务，确保任务不偏离教学重点；设计测试用例时，覆盖“基础场景（验证知识点掌握）”“边界场景（验证逻辑严谨性）”“异常场景（验证代码健壮性）”，如“函数参数传递”任务中，测试用例包含“正常参数（int 型）”“边界参数（0、最大值）”“异常参数（负数、字符型）”；为每个任务配套“知识点回顾链接”“同类例题参考”“调试技巧文档”，如“指针任务”配套“指针与数组关联讲解视频”“指针常见错误调试步骤”，帮助学生自主解决问题。

3.4.2 学习引导者个性化干预与支持

每周分析平台数据，识别三类学生并针对性干预。进度滞后学生（未完成基础级作业）通过平台私信发送“个性化学习建议”，推荐“基础知识点微课”，安排 1 对 1 答疑；反复出错学生引导学生梳理“错误日志”，分析根本原因（如“是否未理解指针地址概念”），补充专项练习；能力突出学生推荐“C 语言竞赛题库”“开源项目贡献”等拓展方向，提供进阶指导；对自动评测无法覆盖的“代码创新性”“逻辑优化思路”，每周选取 10-15 份典型作业（含优秀作业、问题作业），进行人工点评，标注“亮点（如‘用函数封装重复逻辑，代码复用性高’）”“改进方向（如‘可通过宏定义简化常量使用’）”，并在平台展示；在平台创建“C 语言学习社区”，教师定期发布“编程技巧分享”“常见问题解答”，引导学生提问交流，对学生提出的“自动评测未覆盖的问题”（如“代码可读性与效率的平衡”），组织讨论并总结结论。

4 在线编程平台自动评测功能对 C 语言学习效率的提升措施实施关键保障

4.1 平台功能适配保障

选择支持“自定义错误提示”“多维度评分”的平台（如 PTA 高级版、自定义 OJ 系统），确保措施中“精细化反馈”“规范性评分”可落地；平台需具备“学生学习行为统计”（如登录次数、作业完成时长、错误类型分布）、“任务完成情况导出”功能，为教师干预提供数据支撑；确保平台支持 C 语言标准语法（如 C99、C11），兼容常见编译器（如 GCC），避免因平台兼容性问题导致“正确代码评测错误”，影响学生学习体验。

4.2 教师能力提升保障

组织教师参加“自动评测平台高级功能使用”“测试用例设计技巧”培训，提升平台操作与任务设计能力；定期开展“自动评测教学案例分享会”，教师交流“分层任务设计经验”“错误提示优化方法”，形成良性互动；联合计算机专业与教育技术专业教师，共同研究“技术工具与教学规律的融合点”，优化措施设计，提升科学性。

4.3 学生适应引导保障

开学第 1 周开展“自动评测平台使用培训”，讲解“代码提交流程”“错误提示解读方法”“错题集使用技巧”，降低学生使用门槛；引导学生建立“编程日志”，记录“每日任务完成情况”“错误修改过程”“知识点总结”，结合自动评测反馈，形成“学习-反思-提升”闭环；设立“自动评测之星”“最佳代码奖”，对“完成挑战级任务”“代码规范性优秀”“帮助同学解决错误”的学生给予表彰，激发学习积极性。

5 结论

自动评测功能需通过“全流程措施嵌入”实现价值最大化：仅作为“作业批改工具”无法充分发挥其作用，需结合课前预习、课中互动、课后强化，构建“理论-实践-反馈-巩固”的完整学习链；措施设计需紧扣 C 语言学习规律与学生需求：分层任务解决“实践同质化”问题，精细化反馈解决“纠错低效”问题，教师引导解决“能力转化薄弱”问题，三者协同可有效提升学习效率；教师角色转型是措施落地的核心：自动评测功能并未替代教师，而是要求教师从“重复批改”中解放，聚焦“任务设计、数据分析、个性化引导”，成为学生学习的“精准支持者”。

参考文献：

[1] 杨兰,刘伟强,李国强.AI 驱动的 C 语言项目化教学体系重构[J].公关世界,2025,(15):187-189.

[2] 张峰,张红荣.AI 赋能高职 C 语言程序设计课程教学创新——以“函数定义与调用”为例[J].计算机教育,2025,(07):74-79.