基于大模型的 C 语言课程教学改革探索与实践*

刘晋德** 张江江 李诗珂

山西大学计算机与信息技术学院，太原 030006

摘 要 随着国家教育数字化战略的推进，大语言模型（LLM）正重塑计算机基础教育。针对 C 语言课程长期存在的知识抽象、反馈滞后及评价单一等困境，本研究立足应用型高校，提出一种融合建构主义的“人机协同”教学新范式。研究构建了“师-生-机”三元互动的教学生态，通过引入提示词工程、建立 AI 分阶引导与即时反馈机制，并重构以过程交互为核心的评价体系，实现了对传统教学的系统性重塑。基于 2024 级计算机专业 86 名学生的双盲实验数据表明，该方案不仅使期末不及格率同比下降 8.5%，更显著提升了学生解决复杂工程问题的逻辑能力与代码产出。研究认为，将大模型转化为“数字助教”与“认知脚手架”，能有效降低认知负荷，推动学生从代码工匠向具备高阶计算思维的系统设计者转型，为人工智能时代的编程教育改革提供了可复制的实践路径。

关键字 大语言模型，C 语言，人机协同，计算思维

Exploration and Practice of C Language Course Teaching Reform Based on Large Models

Jinde Liu Jiangjiang Zhang Shike Li

College of Computer and Information Technology, Shanxi University
Taiyuan 036006, China

Abstract—Driven by the advancement of the national education digitalization strategy, Large Language Models (LLMs) are reshaping fundamental computer education. Addressing persistent challenges in C language courses—such as high knowledge abstraction, delayed feedback, and singular evaluation frameworks—this study proposes a novel "human-machine collaborative" teaching paradigm integrating constructivist theory, tailored for applied universities. The research constructs a "Teacher-Student-Machine" ternary interactive ecosystem. By introducing Prompt Engineering, establishing AI-based phased guidance with real-time feedback, and reconstructing an evaluation system centered on process interaction, the study achieves a systemic reshaping of traditional teaching. Data from a double-blind experiment involving 86 Computer Science students from the Class of 2024 indicate that this approach not only reduced the final exam failure rate by 8.5% year-on-year but also significantly enhanced students' logical capabilities and code output in solving complex engineering problems. The study concludes that transforming large models into "digital teaching assistants" and "cognitive scaffolds" effectively lowers cognitive load and facilitates the student's transition from a mere code artisan to a system designer equipped with higher-order computational thinking, providing a reproducible practical path for programming education reform in the AI era.

Keywords—Large Language Models, C Language Teaching, Human-Machine Collaboration, Computational Thinking

1 引 言

1.1 研究背景：数字浪潮下的教育

当前，新一轮科技革命和产业变革加速演进，教育数字化已不再是单纯的“将黑板搬到屏幕上”的技术修补[1]，而是关乎人才培养模式根本性转型的国家战略行动[2]。教育部在教育数字化战略行动中明确提出，要利用智能技术支撑人才培养模式的创新[3]。

在这一宏观背景下，计算机基础教育正处于风口浪捷。2023 年以来，大语言模型（LLM）的爆发式增长，展现出了惊人的代码生成、逻辑解释与错误诊断能力。对于一线教师而言，这不仅是技术的迭代，更是一场不得不面对的危机——当学生只需将题目复制粘贴给 AI 就能瞬间获得完美运行的代码时[4]，我们沿用了几十年的编程教学逻辑是否已经失效？

C 语言以其严谨的语法结构、精妙的指针操作和底层的内存管理机制，历来是高校理工科学生构建计算机系统观、培养严密逻辑思维的首选入门语言。然而，技术的便利性正在消解学习过程的艰巨性。我们无法固步自封，简单的物理屏蔽（如断网考试）只能延缓问题的暴露，却无法阻挡技术渗透的趋势。因此，如何变“堵”为“疏”，在承认大模型强大的前提下，重新定义 C 语言课程的教学目标与核心价值，已成为每一位计算机教育工作者亟待解决的课题。

基金资助：本文得到山西大学教改项目“教育数字化背景下的基于大模型的 C 语言课程教学改革探索与实践”支持。
**通讯作者：苟仲文 csteic3@163.com

1.2 研究意义：从教会语法到重塑思维

本研究的意义在于试图回答一个核心问题：在 AI 能够通过图灵测试甚至在编程能力上超越初级程序员的时代，我们为什么还要教学生手写 C 语言？

传统的 C 语言教学往往侧重于语法细节的记忆（如运算符优先级、格式化输出符等），这种以知识点为中心的教学在 AI 时代显得性价比极低。本研究通过引入大模型作为教学要素，旨在探索一种从语法驱动向思维驱动转型的路径。我们希望证明，AI 不仅不是学生思维的替代品，反而是训练学生计算思维、提升复杂系统设计能力的最佳陪练。通过教学改革，我们不再致力于培养能默写代码的人形编译器，而是培养能够驾驭 AI、利用 C 语言原理解决实际工程问题的架构师[5]。

2 理论基础与教学观重构

2.1 建构主义学习理论的现代化演绎

皮亚杰（Piaget）的建构主义理论认为，知识不是通过教师传授得到的，而是学习者在一定的情境下，借助他人（包括教师和学习伙伴）的帮助，利用必要的学习资料，通过意义建构的方式获得的[6]。

在传统的 C 语言课堂中，这个“情境”往往是枯燥的 IDE 界面，“伙伴”仅限于邻座同学。引入大模型后，AI 实际上扮演了一个全知且全天候的“高能伙伴”。学生在与 AI 进行多轮对话、调试代码的过程中，实际上是在不断地同化与顺应新知识。例如，当学生发现 AI 生成的代码在特定边界条件下出错时，他必须调用自己对 C 语言内存机制的理解去修正它，这一过程正是深度的知识建构。

2.2 认知负荷与知识联通

维果茨基（Vygotsky）提出的“最近发展区”理论强调教学应走在发展的前面。对于大一新生而言，C 语言的指针、链表等概念往往超出了其现有的认知舒适区，导致认知负荷过载（Cognitive Overload）。在传统教学中，教师作为唯一的知识提供者，精力极其有限。大模型作为一种“数字脚手架”，具有极强的适应性：对于基础差的学生，它可以提供逐行注释、类比解释（如将指针比作门牌号），降低认知门槛。对于优等生，它可以提供代码优化建议、算法变体，拓展认知的上界。 这种动态调整的支撑能力，使得大规模因材施教成为可能。在数字化时代，知识的管道比知识的内容更重要。C 语言教学改革的核心，不再是让学生死记硬背 stdlib.h 中有多少个函数，而是建立“问题-逻辑-工具-解决方案”的连接。大模型正是那个连接器，教会学生如何快速检索、验证并整合外部知识，是本轮改革的核心方法论。

3 C 语言课程教学现状深度剖析

3.1 情画像

目前的授课对象主要是大一新生。经过多年的一线观察，我们发现这一代学生具有鲜明的群体特征：具象思维与抽象逻辑的断层，他们习惯了图形化界面（GUI）和所见即所得的操作，对 C 语言这种“黑框框”里的命令行交互、内存地址操作感到极度陌生。从高级语言的封装思维下沉到 C 语言的底层裸机思维，存在巨大的认知鸿沟。短视频时代的耐心缺失，长期受碎片化信息流投喂，学生对长逻辑链条的专注力下降。C 语言中一个 Bug 可能需要数小时的排查，这种延迟满足感极易导致学生在入门期产生习得性无助。两极分化加剧，在同一个行政班级里，有的学生高中已通过 NOIP（信息学奥赛），觉得课程进度太慢；而有的学生连基本的键盘盲打都很困难。这种方差极大的生源结构，使得统一节奏的课堂教学常常顾此失彼[7]。

3.2 传统教学痛点

编程是一门实践性极强的学科，其核心在于“试错-反馈-修正”的闭环。然而，现有的教学资源配置导致这一闭环经常断裂：实验课的保姆式指导：目前高校普遍的大班授课制（往往 100 人以上大班，实验课 40-50 人），师生比严重失调。在实验课上，教师往往化身为人肉调试器，疲于奔命地帮学生解决漏了分号、全角符号等低级错误。深层逻辑指导缺位：由于时间被低级错误挤占，教师很难有精力去引导学生思考为什么这里用链表比数组好、这个递归会不会爆栈等高阶问题。学生往往在改掉语法错误后就停止思考，代码虽然跑通了，但逻辑是一团乱麻[8]。

3.3 新型危机：大模型背景下的假性学习

这是目前最为棘手的问题。大模型的出现，对现有的作业与考试模式构成了降维打击。作业功能的失效：传统的编程作业多为算法题（如水仙花数、斐波那契数列）[9]。现在的学生只需将题目复制给 AI，AI 能在一秒钟内给出包含注释的完美代码。“代码跑得通，原理讲不通”：我们发现，部分学生在提交了完美作业后，面对面询问代码中某一句 p->next 的含义时，竟然一脸茫然。这种外包式学习如果得不到干预，将导致整整一代计算机学生丧失核心竞争力——他们将沦为 AI 的搬运工，而非控制者。

面对上述现状，简单的封堵（如禁止使用 AI）已被证明是徒劳的。我们必须承认，AI 已经成为编程生态的一部分。改革的唯一出路，是将 AI 从作弊工具合规化为“教学工具”，将其纳入课程体系的正规军。

4 师-生-机三元协同教学模式的构建

面对 AI 技术的强势介入，简单的“封堵”策略已被证明行不通 。C 语言教学改革的核心不应是回避工具，而是重新定义人与工具的关系，构建一种融合、共生的新型教学形态。我们提出将传统的“教师-学生”二元结构扩展为“教师-学生-机器（大模型）”三元生态系统，详细结构如图 1 所示。

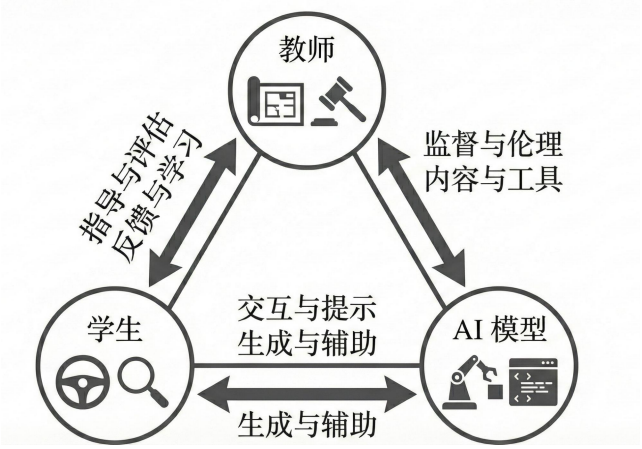

图 1 三元协同教学模式

4.1 角色重塑：从二元对立到三元共生

在这一新生态中，三者的功能定位发生了根本性迁移，教师从“知识搬运工”转变为“教学架构师”与“价值引领者” 在传统模式下，教师耗费大量精力讲解 if-else 嵌套、for 循环语法规则，而学生往往知其然不知其所以然。在大模型背景下，语法的记忆与检索成本趋近于零。因此，教师的职能必须“向上迁移”：

（1）场景设计者：设计那些 AI 难以直接生成完美答案的、具有高阶思维挑战的复杂工程案例。

（2）仲裁者与审稿人：教会学生如何鉴别 AI 生成代码的优劣，指出 AI 在内存安全、算法效率上的潜在缺陷。

（3）伦理引导者：在技术狂欢中保持冷静，向学生灌输学术诚信、知识产权保护及技术使用的边界意识。

学生从“代码录入员”转变为“人机协作驾驶员”与“代码审核员” 学生不再是被动的知识容器。在拥有了 AI 这一强力外脑后，学生的主体性体现在提问与判断上。他们需要学习如何用精准的自然语言描述编程需求（Prompting），并具备足够的理论基础去 Review(审查)AI 生成的代码。能够读懂并修改 AI 的代码将取代从零手写每一行代码，成为新的核心能力标准。

大模型（The Copilot & Scaffold）：从作弊工具转变为全天候数字助教，大模型在教学中应当被定义为全能助手。它承担以下功能：

（1）个性化答疑：解决大班教学中教师无法顾及每一位学生细微疑问的痛点。

（2）样板生成：为学生提供不同风格的代码实现（如分别用数组和指针实现同一功能），以此拓宽学生的视野。

（3）情绪价值：作为不厌其烦的陪练，它能显著降低后进生因频繁报错产生的挫败感。

4.2 三元互动运行机制

在该模式下，教学流程不再是线性的“讲授-练习-批改”，而是循环迭代的：教师发布任务（强调逻辑设计与系统架构）；学生拆解任务，与 AI 协作生成基础模块；学生整合并调试代码，利用所学知识修正 AI 的逻辑谬误；师生共同评审，教师针对人机协作的过程进行点评。

该模式通过构建动态迭代的反馈闭环，显著深化了学生的认知层级。在人机协作中，学生需通过鉴别与修正来验证 AI 产出，迫使其从机械的代码编写者转型为具备批判性思维的系统架构师。同时，评价重心由结果转向协作过程，使教师能精准定位并干预学生的逻辑短板。这种机制不仅有效提升了教学效率，更让学生提前适应了人机共生的现代工程开发环境，强化了解决复杂问题的实战能力。

5 改革实施路近与具体措施

基于上述理念，我们在计算机科学与技术专业的 C 语言课程中实施了以下全生命周期的改革措施。我们力求将大模型深度嵌入课前、课中、课后的每一个毛细血管。整体的改革框架如图 2 所示。

5.1 课程内容重塑

这项工作的核心要点在于引入提示词工程即 Prompt Engineering[10]与代码鉴赏。我们不得不承认，教学生如何问问题在当下比如何写分号更重要。因此，我们在教学大纲中增设了 4 学时的 AI 辅助编程导论模块 。

（1)提示词工程（Prompt Engineering）实战训练

我们将计算思维具象化为提示词构建能力。在课堂上，我们展示了低效提问与高效提问的对比，训练学生将模糊需求转化为结构化指令[10]。

反面教材：帮我写一个学生管理系统。这样做的结果 AI 会生成一个通用的、简单的代码，学生直接复制，完全不理解内部逻辑，且可能包含未学的语法（如结构体早于教学进度出现）。

正面教材：我正在学习 C 语言的结构体和文件操作。请帮我设计一个学生管理系统的代码框架。要求：使用单链表存储数据；包含增加、删除、查找功能的函数原型声明；不要直接给出完整代码，而是解释每个函数的设计思路；指出在这个设计中，内存管理需要注意哪些潜在风险。

通过这种训练，学生被迫在提问前先厘清数据结构与算法逻辑，这本质上是一次高强度的思维预演。

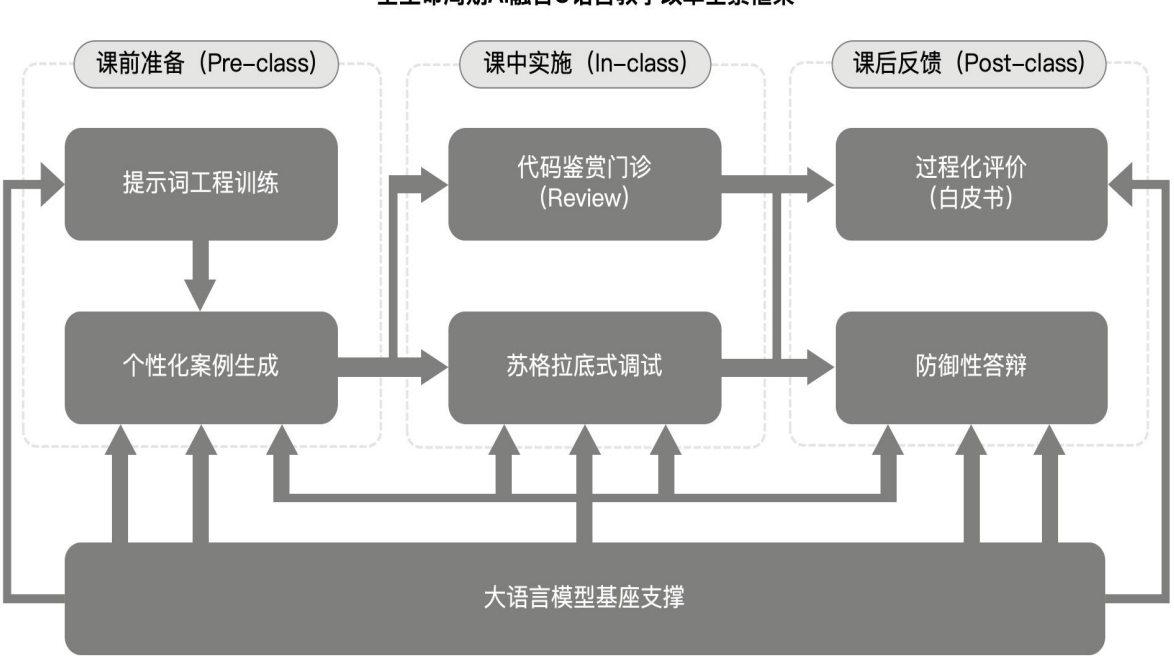

图 2 C 语言教学改革全景框架

（2)逆向教学法：代码鉴赏与找茬门诊

利用大模型偶尔产生的“幻觉”或非最优解，我们设计了“代码门诊（Code Clinic）”环节 。

教学案例：在讲解《指针与内存管理》章节时，我们故意使用 AI 生成一段看似正确但存在内存泄漏（Memory Leak）或缓冲区溢出（Buffer Overflow）的代码。一段使用了 malloc 但未进行 free，或者在 strcpy 时未检查目标数组长度的代码。如下代码所示。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
    /**
     * AI 生成的注释：
     * 该函数接收一个源字符串，动态分配内存并返回副本。
     * 使用 malloc 确保内存空间刚好足够容纳源字符串。
     */
char* create_string_copy(const char* src) {
    if (src == NULL) return NULL;

    // [陷阱重点]
    // AI 逻辑：源字符串有多长，我就申请多大的空间
    // 错误：忽略了字符串末尾的 '\0' 结束符需要占用 1 个  字节
    char* dest = (char*)malloc(strlen(src));
    if (dest != NULL) {
        // [崩溃触发点]
        // strcpy 会尝试写入 '\0'，导致向 dest 分配的内存块
        // 之后的第 1 个字节写入数据，破坏堆内存结构
        strcpy(dest, src);
    }

    return dest;
}

int main() {
    char*  my_copy  =  create_string_copy("Hello,  AI World!");
    if (my_copy) {
        printf("Copy success: %s\n", my_copy);
        free(my_copy);
    }
    return 0;
}
```

活动设计：要求学生扮演“资深工程师”，对这段 AI 代码进行 Code Review。学生需要指出错误所在的行数，解释后果（如导致程序崩溃或内存耗尽），并给出修正方案。

成效：这种“找茬”带来的成就感远超枯燥的语法填空，极大地提升了学生对 C 语言底层机制的敏感度。

5.2 实验教学改革：分阶引导与及时反馈

针对实验课指导不过来的顽疾，我们设计了基于 AI 的分阶实验流程，核心原则是：AI 只能作为导航员，不能作为代驾。

（1）预习阶段：

个性化案例生成 C 语言的难点在于抽象。不同学生对抽象概念的理解路径不同。对于数学基础好的学生，引导其让 AI 用数学公式解释“递归”；对于文科思维较重的学生，引导其让 AI 用“俄罗斯套娃”或“排队报数”的生活隐喻来解释同个概念。这种个性化的预热，使得学生在进入实验室前已建立了初步的心理表征。

（2）实验阶段：

限制性交互与苏格拉底式调试，我们在实验室推行了严格的“5 分钟思考原则”：遇到 Bug，必须先自己排查 5 分钟。若无果，方可求助 AI，但必须遵守“禁止直接索要代码”的铁律。

我们规定了标准的提问范式（Template）：“我的代码在第 X 行遇到了报错 [错误信息]。我目前的理解是 [自己的分析]。请不要直接给我正确的代码，而是告诉我：这个错误的原理是什么？我应该通过什么思路去调试它？”这种机制迫使学生必须阅读报错信息（Compiler Warnings），并进行初步的逻辑假设。AI 在此过程中仅充当“提示者”，引导学生自己通过断点调试找到问题，从而保护了宝贵的试错体验。

5.3 评价体系重构

这是改革中最艰难也最关键的一环。既然代码结果（Product）可以轻易获取，评价的锚点必须转移到思维过程（Process）和答辩能力（Defense）上 。

（1）过程化评价的权重提升

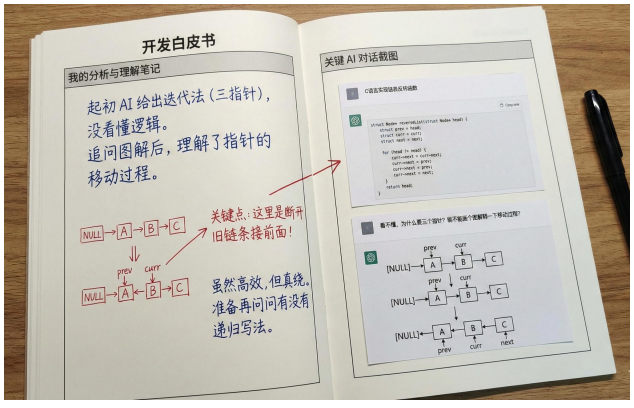

图 3 白皮书

我们大幅调整了总评成绩的构成：平时作业代码运行结果的权重从 50% 降至 20%，而新增了 30% 的人机协作日志权重 。学生在提交大作业（如课程设计）时，必须附带一份《开发白皮书》。其中必须包含关键的 AI 对话截图。例如：“在实现链表反转时，AI 给出的代码使用了三个指针，我没有看懂，于是我追问了它图解过程，并最终发现可以用递归实现，以下是我的对比分析……”图 3 展示了我们使用的白皮书。对于评分标准，教师不只看代码是否跑通，更看学生是否主导了与 AI 的对话，是否展现出了批判性思维。

（2）引入防御性答辩机制

为了彻底杜绝假性学习，我们在期末引入了类似研究生答辩的口试环节。教师在学生的作业代码中随机选中一行（尤其是复杂的指针运算或位操作），问：“如果把这行代码删掉，程序会发生什么？”或者“为什么要在这里加 const 修饰符？”要求学生现场修改一个参数或逻辑。例如：“现在把这个单向链表改成双向链表，你需要改动哪几处？” 这是检验学生是否真正掌握知识的“试金石”——AI 可以替写代码，但无法通过脑机接口替学生回答现场的逻辑质询。

（3）确立 AI 使用边界与学术诚信

我们在第一节课就与学生签署《课程 AI 使用诚信公约》 。公约明确界定合规行为包括：用 AI 解释概念、生成测试数据、查找语法错误、优化变量命名。违规行为包括：直接复制 AI 代码作为作业提交、无法解释自己提交的代码逻辑。 通过这种仪式感的建立，引导学生树立负责任的人工智能使用观[11][12]。

6 实施效果与成效评估

为了客观评估“师-生-机”三元协同模式的教学实效，我们以我校计算机科学与技术专业 2024 级两个试点班（共 86 人）为实验组，选取同专业且入学成绩无显著差异的两个平行班（共 84 人）为对照组，开展了为期两个学期的纵向对比研究 。评估采用定量数据分析（成绩与代码量）与定性反馈（问卷与访谈）相结合的混合研究方法。

6.1 量化数据分析

（1）成绩分布形态的显著变化

传统的 C 语言期末成绩通常呈正态分布，低分段学生往往占据一定比例。然而，在引入大模型辅助教学后，试点班的成绩分布呈现出明显的“负偏态”特征（即高分段人数增加）。

如表 1 所示，对照组的卷面不及格率为 14.2%，而试点班仅为 5.7%，同比下降了约 8.5 个百分点 。这一数据有力地证明，大模型作为全天候助教，在解决指针理解障碍、内存错误排查等基础痛点上，对后进生起到了关键的兜底作用，不及格率大幅下降。

试点班 85 分以上学生的比例达到 32%，高于对照组的 19%。这表明 AI 并没有拉平差距，而是帮助头部学生突破了天花板，优秀率有一定提升。如表 1 所示。

（2）代码产出量与复杂度的倍增

为了衡量学生的实际编程投入度，我们统计了两个组别在期末综合课程设计环节的代码提交情况。试点班学生提交项目的平均代码行数为 680 行，是对照组（425 行）的 1.6 倍 。以往大一新生的课设题目多局限于“学生成绩管理系统（控制台版）”等增删改查类项目。而在试点班，约 45% 的学生尝试了涉及图形库（如 EasyX）、多文件编译甚至简单网络通信的项目。例如，“贪吃蛇游戏”的 AI 对战版、“简易通讯录”的云同步功能等，这些在传统教学中被视为超纲的内容，如今在 AI 的辅助下成为了学生的普遍实践 。

6.2 问卷调查反馈

期末我们向试点班发放了《AI 辅助 C 语言学习体验调查问卷》，回收有效问卷 86 份 。92% 的学生表示，大模型显著降低了编程初期的焦虑感与挫败感 。一位学生在开放性问题中写道：“以前遇到 Segmentation Fault 查不出原因想砸键盘，现在把错误代码丢给 AI，它能告诉我大概率是指针越界，这种即时反馈是救命稻草。”78% 的学生认为，通过反复修改 Prompt 与 AI 对话，自己学会了如何将一个模糊的需求拆解为具体的函数模块。这种计算思维的习得，比单纯记忆语法更有价值。约 30% 的学生反映，初期曾被 AI 生成的错误代码（如引用了不存在的库函数）误导，浪费了大量调试时间 。但这恰恰成为了我们教学的契机——这部分学生在后续的学习中展现出了更强的代码审查（Code Review）意识，他们不再盲信权威，而是学会了验证。

6.3 典型案例研究

在期末答辩环节，一名入学时编程基础薄弱的学生的案例极具代表性。 他提交了一个“基于文件操作的图书管理系统”。在答辩中，教师质疑其核心排序算法的来源。他坦诚地回答：“老师，快排（Quick Sort）算法确实是 AI 生成的。” 但他随即补充道：“最初 AI 给的是冒泡排序，我在测试 10 万条数据时发现太慢，于是我问 AI 有没有更快的算法。它给出了快排，但原代码中的递归深度太深导致了栈溢出。我又查阅资料，让 AI 帮我改成了非递归版本，并手动优化了其中的交换逻辑以适配我的结构体定义。” 在这个案例中，这位同学虽然没有亲手写出每一行排序代码，但他完成了性能测试、算法选型、错误修正、代码适配的全过程。这正是我们所追求的——学生从底层的代码搬运工，晋升为了系统的架构师 。

7 结束语

本研究证实，生成式 AI 的引入并非对传统 C 语言教学的颠覆，而是对其“回归本质”的深刻重构。通过构建“师-生-机”三元协同的教学生态，研究有效化解了规模化教学与个性化指导之间的结构性矛盾，成功推动了人才培养标准从低阶的“语法记忆”与“编码速度”向高阶的“问题拆解”、“提示词工程”及“系统调试”能力跃迁。实践表明，基于“限制性交互”规则与过程化评价的教学改革，能显著提升学生解决复杂工程问题的能力，为理工科课程的数智化转型提供了可复制的范式[7][1]。

然而，教学改革的深水区仍面临挑战。如何厘清人机协作中“辅助”与“替代”的边界以规避思维惰性，如何应对技术迭代带来的“红皇后效应”，以及如何填补算力差异导致的数字鸿沟，仍需在长期的教学实践中持续求解。

表 1 试点班与对照班期末成绩分布

指标维度	成绩/项目区间	对照班（N=84，传统模式）	试点班（N=86，AI协同模式）	同比变化
成绩分布	不及格（<60分）	12 人（14.3%）	5 人（5.8%）	下降 8.5%
	中等（60-84分）	56 人（66.7%）	54 人（62.8%）	持平
	优秀（≥85分）	16 人（19.0%）	27 人（31.4%）	提升 12.4%
能力指标	平均代码行数	425 行	680 行	提升 60%
	复杂项目占比	15.50%	45.30%	提升 29.8%
	调试排错平均耗时	45 分钟/次	18 分钟/次	效率提升 2.5倍

展望未来，编程教育应将大模型视为与编译器同等重要的基础设施。后续研究将致力于研发垂直领域的教学大模型，通过微调技术实现从“直接生成”向“引导启发”的功能迭代。教师的使命终将从传授代码

规范演变为培养能够驾驭 AI 的系统设计者，这不仅是技术演进的必然选择，更是工程教育回应智能化时代需求的必由之路。

参 考 文 献

[1] 董晓晓, 吴砥. 生成式人工智能赋能教育数字化转型的机遇、挑战与路径[J]. 中国电化教育, 2023(7): 12-19.

[2] 谭貌, 段斌, 周彦, 等. 面向产出落实工程教育认证标准的院系机制与实践[J]. 计算机技术与教育学报, 2023, 11(5): 16-20.

[3] 裴壮, 田秀霞, 李冰雪. 知识图谱赋能的面向对象程序设计C++教学改革与实践[J]. 华东师范大学学报(自然科学版), 2024(5): 104-113.

[4] Prather J, Denny P, Leinonen J, et al. The Scaffolding Role of Large Language Models in Introductory Programming Education[J]. ACM Transactions on Computing Education, 2024, 24(1): 1-25.

[5] 刘敏, 李杨. 生成式人工智能对大学生计算思维培养的影响研究[J]. 现代教育技术, 2024, 34(2): 15-22.

[6] 方维, 袁宝库, 梁峰绮. 基于PTA平台的程序设计类课程教学改革实践[J]. 计算机技术与教育学报, 2022, 10(1): 97-100.

[7] 李永庆, 孙丽敏, 孙媛媛, 等. 国产化背景下C语言课程建设与创新实践[J]. 计算机技术与教育学报, 2024, 12(5): 19-23.

[8] 柯毅明, 汤宏誉, 刘敏. 基于OBE和CDIO的C语言程序设计课程教学改革[J]. 计算机教育, 2024(2): 20-25.

[9] 王烨, 张培欣. 计算机程序设计与信息安全数学基础课程协同建设初探[J]. 计算机技术与教育学报, 2024, 12(2): 49-55.

[10] White J, Fu Q, Hays S, et al. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT[J]. arXiv preprint arXiv:2302.11382, 2023.

[11] 傅继彬. 构建计算机网络课程中的思政教育协议栈[J]. 计算机技术与教育学报, 2022, 10(5): 23-26.

[12] 李海峰, 王炜. 生成式人工智能在教育领域的应用伦理风险与治理路径[J]. 现代教育技术, 2023, 33(10): 14-22.