程序设计课程教学双轮驱动新模式探索

傅佑铭 梁超 何政 刘峰

武汉大学计算机学院，武汉 430000

摘 要 针对传统的程序设计(以 C++为例)课程教学中存在的重理论轻实践、方法单一、知识点难以全覆盖等问题，本文提出"贯穿式案例+AI 大语言模型"双轮驱动的创新教学模式。该模式以学生成绩管理系统为工程化实践主线，分阶段融入 UML、类、继承、多态、泛型和模板等核心知识点。同时摒弃传统的以 PPT 讲述为主的教学方式，借助主流 AI 大语言模型等辅助软件工具实现代码智能纠错、个性化学习路径推荐及实时互动答疑，同时也针对性地预设了 AI 路径依赖规避机制，引导学生注重自身实际能力的提升。教学实践表明，该创新教学模式有效帮助学生理解课程概念原理，有效提升编程水平、逻辑思维和架构设计等计算机基本技能。学生的作业代码质量与问题解决能力显著增强。

关键字 程序设计，贯穿式案例，去 PPT 化，AI 大语言模型

New Dual-Wheel Drive Teaching Model for Programming Courses

Fu Youming Liang Chao He Zheng Liu Feng
School of Computer Science, Wuhan University
Wuhan 430000, China
fuym@whu.edu.cn

Abstract—In response to issues in traditional programming course instruction (using C++ as an example), such as emphasizing theory over practice, monotonous teaching methods, and difficulties in covering all key knowledge points comprehensively, this paper proposes an innovative teaching model driven by a dual approach: a "threaded case + large AI language models." This model uses a student performance management system as the main engineering practice thread, gradually incorporating core knowledge points such as UML, classes, inheritance, polymorphism, generics, and templates. Simultaneously, it abandons the traditional PPT-dominated teaching approach and leverages mainstream large AI language models and other auxiliary software tools to achieve intelligent code error correction, personalized learning path recommendations, and real-time interactive Q&A. Additionally, an AI path dependency avoidance mechanism is deliberately 预设 to guide students in focusing on improving their practical skills. Teaching practice shows that this innovative model effectively helps students understand course concepts and principles, significantly enhancing their programming proficiency, logical thinking, and foundational computer skills such as architectural design. The quality of students' assignment code and problem-solving abilities has markedly improved.

Keywords—programming, threaded case, de-PPTization, large language models

1 引 言

程序设计语言在计算机科学与技术领域占据着举足轻重的地位。其中的主流代表是C++语言[2][4][6]，不仅继承了 C 语言的高效性和灵活性，还引入了面向对象编程、泛型编程等先进特性，使其在系统软件开发、游戏开发、人工智能、大数据处理等众多领域得到广泛应用。掌握高级程序设计语言编程技能，对于计算机专业学生来说，是开启职业生涯的关键一步，也是深入学习其他高阶计算机课程的重要基础[1][2][3]。

本课程教学创新旨在探索一种创新的高级语言程序设计课程教学模式，将一个教学实践案例贯穿始终，并结合主流 AI 大语言模型，提升学生的学习效率和学习质量，为学生后续学习高阶课程（如数据结构、操作系统等）打下坚实的基础，同时培养学生扎实的计算机基本技能，提高学生的综合素质和就业竞争力[4][5]。

2 教学现状分析

论文题目、作者、摘要、关键词要有中文和英文。在传统的高级程序设计语言教学中，存在着诸多问题，严重影响了教学效果和学生的学习体验[6]，例如：

（1）重理论轻实践：教学过程往往侧重于高级程序设计语言语言的语法规则、概念原理等理论知识的传授，而对实践环节的重视程度不足。使得学生在学习过程中缺乏足够的实践机会，难以将所学的理论知识转化为实际的编程能力[7][8]。

（2）教学方法单一：主要以教师讲授为主，采用“满堂灌” 的教学方式。教师在课堂上通过板书或 PPT 演示，向学生讲解高级程序设计语言的知识要点，学生则被动地接受知识。这种教学方法缺乏互动性和趣味性，难以激发学生的学习兴趣和主动性[9]。

（3）教学内容难以全覆盖：C++语言是一门动态

演变的语言，从1998年ISO C++标准委员会发布第一个C++标准至今已经历多个版本且同时被业界应用，相应的各类IDE工具也不断推陈出新与时俱进，对应的教学知识点也在不断积累的更新，单纯依靠课程教学基本无法做到绵绵覆盖[9][10]。

（4）考核方式单一：通常以期末考试成绩为主，辅以平时作业和考勤。这种考核方式注重对学生理论知识的考查，而对学生的实践能力、创新能力等方面的考核不足。在期末考试中，往往以选择题、填空题、简答题等形式考查学生对高级程序设计语言语法和概念的记忆，而对学生的编程实践能力考核效果有限。这导致学生为了取得好成绩，往往死记硬背理论知识，而忽视了对实际编程能力的培养[11][12]。

3 创新教学举措

论文题目、作者、摘要、关键词要有中文和英文。在传统的高级程序设计语言教学中，存在着诸多问题，严重影响了教学效果和学生的学习体验[5]，例如：

（1）重理论轻实践：教学过程往往侧重于高级程序设计语言语言的语法规则、概念原理等理论知识的传授，而对实践环节的重视程度不足。使得学生在学习过程中缺乏足够的实践机会，难以将所学的理论知识转化为实际的编程能力[7]。

3．1 中使用贯穿式教学案例

针对上述教学中存在的问题，笔者在教学实践中选择一个综合性强、趣味性高且贴近实际应用的案例，如开发一个简单的学生成绩管理系统。该系统可涵盖变量、数据类型、控制结构、函数、类与对象、继承、多态、文件操作等 C++ 核心知识点，然后按教学大纲按如下步骤分阶段推进教学工作。

（1）基础语法阶段。此阶段着重讲解 C++ 基础语法，如变量定义、数据类型、控制结构。让学生实现成绩录入和简单输出功能，定义学生类的基本属性，如姓名、学号、各科成绩等，并学习使用标准输入输出方法。在这一阶段，学生可以初步了解 C++ 的基本结构和语法，为后续学习打下基础。这一阶段的教学重点是让学生熟悉C++的基本语法和编程环境，培养学生的编程兴趣和初步的编程能力。

（2）函数与类阶段。引入函数和类的概念后，将成绩录入、输出等功能封装成函数，完善学生类，添加构造函数、析构函数和成员函数，如计算平均分、判断是否及格等。在这一阶段，学生可以学习如何将程序分解为多个函数和类，提高代码的可读性和可维护性。通过定义类和成员函数，学生可以更好地理解面向对象编程的基本概念，如封装、抽象等。例如，学生可以学习如何定义一个学生类，包含学生的姓名、学号、成绩等属性，以及计算平均分、判断是否及格等方法。通过这种方式，学生可以更好地理解如何将现实世界中的对象抽象为类。

（3）继承与多态阶段。讲解继承和多态知识时，对学生类进行扩展，创建不同类型的学生类（如本科生、研究生），并实现多态的成绩显示功能。在这一阶段，学生可以学习如何通过继承来扩展类的功能，以及如何使用多态来实现灵活的程序设计。同时学生可以学习如何使用继承来共享代码和实现多态。通过多态，学生可以编写更加灵活和通用的代码，提高程序的可扩展性和可维护性。

（4）泛型和模板阶段。引入泛型编程和模板的概念，让学生使用模板来实现更通用的功能。例如，实现一个通用的成绩排序模板函数，该函数可以对不同类型的学生成绩容器进行排序。在这一阶段，学生可以学习如何使用模板来编写通用的代码，提高代码的复用性和灵活性。通过使用模板，学生可以编写出更加通用和灵活的代码，提高程序的可扩展性和可维护性。

（5）文件和数据库操作阶段。最后教授文件操作，让学生实现学生成绩信息的保存和读取功能，将学生成绩信息存储到文件中，并在程序启动时从文件中读取信息。在这一阶段，学生可以学习如何将程序与外部数据存储进行交互，提高程序的实用性和可扩展性。

在上述每个阶段，教师会加强对学生实践的指导，及时解答学生遇到的问题。同时，设计合理的考核方式，如阶段小测验、最终项目展示等，检验学生对案例的掌握程度。

相比于传统教学方法，笔者采用上述贯穿式教学案例有如下成效：

（1） 提升学生认知框架构建过程的完整性。传统教学按不同章节和知识点给出零散案例，学生难以将所学知识串联起来。而以一个实践案例贯穿始终并持续迭代，学生能直观清晰看到各个知识点在实际项目中的关联和应用顺序，形成完整的知识体系。

（2）提高学习兴趣和动力，有效消除知识孤岛。单一的综合性案例具有明确的目标和实际应用场景，能激发学生的学习兴趣和探索欲望。例如，在案例项目演进过程中，后续可以用用异常处理机制解决前期面临的学生信息数据输入正确性的校验问题，使用RTTI模型实现通过抽象类型获得具象载体，有效避免学生产生知识点不知如何使用的割裂感。同时，教师可以进行痛点场景植入，故意保留可优化环节，如初期使用裸指针，后期引入智能指针进行对比，使得学生在逐步完成案例的过程中，获得成就感，从而更积极主动地学习。

（3）高效提升知识迁移和工程实践能力。综合性案例要求学生综合运用所学知识解决实际问题，能有效提高学生的编程实践能力和问题解决能力。相比之下，零散案例可能只侧重于某个知识点的练习，学生缺乏整体项目的实践经验。学生在一个完整案例中掌握了知识的综合运用方法，在遇到其他类似问题时，更易于将所学知识迁移过去，提高解决新问题的能力。而传统教学的零散案例可能使学生局限于特定案例，难以灵活运用知识。

表 3.1 贯串式案例教学与传统教学方式对比

维度	传统教学（零散案例）	贯穿式教学（单一主线案例）
知识点关联	碎片化，缺乏系统性	线性递进，展示知识点依赖关系
实践场景	单一功能验证	完整项目生命周期（需求→设计→实现）
知识迁移能力	局限于特定案例	可迁移至同类复杂系统

总之，这种教学法不仅有效提升知识学习效率和留存率，更重要的是能培养出能应对真实复杂系统的全栈式C++工程师思维，使学习者建立起"语言特性-系统设计-业务需求"的三维映射能力。

3．2 AI 赋能的去 PPT 式教学

题目在高级语言程序设计课程中，尤其是C++程序设计的教学中，代码的执行过程和内存管理是学生理解的难点。传统的教学方法往往依赖于静态的PPT演示和教师的口头讲解，难以直观地展示代码运行的动态过程和内存的变化。为了突破这一局限，我们引入了基于AI大语言模型的去PPT式教学方法，通过可视化和动态分析的方式，帮助学生更好地理解和掌握复杂的编程概念。

笔者所使用的程序设计（C++）教学大语言模型工具是一种基于代码运行时信息的动态分析工具，能够实时捕捉和展示代码的执行流程、内存分配与释放、函数调用栈等关键信息。它通过在代码中插入轻量级的追踪点（Tracing Points），收集运行时数据，并以可视化的方式呈现出来。这些数据包括但不限于：1）代码执行路径：展示代码的运行流程，包括分支选择、循环迭代等；2）内存分配与释放：实时显示内存的分配和释放情况，帮助学生理解内存管理机制；3）函数调用栈：展示函数调用的顺序和层次关系，便于理解递归和嵌套调用；4）变量值变化：动态展示变量在运行过程中的值的变化，帮助学生理解变量的作用域和生命周期。

该工具的核心优势在于其动态性和可视化特性，能够将抽象的编程概念具象化，使学生能够直观地观察代码的运行过程，从而加深对编程语言特性和内存管理机制的理解。具体的教学手段包括：

（1）代码执行流程可视化。在C++程序设计教学中，代码的执行流程是学生理解的难点之一，尤其是涉及到复杂的控制结构（如循环嵌套、条件分支）和函数调用时。通过该工具，教师可以在课堂上实时展示代码的执行过程，让学生清晰地看到代码是如何一步步运行的。例如，在讲解递归函数时，教师可以使用工具展示递归调用的深度和每次调用的参数值。学生可以通过观察函数调用栈的变化，直观地理解递归的执行过程，避免了传统教学中仅靠想象和推导的困难。

（2）内存管理可视化。C++语言的内存管理是学生学习的另一个难点，尤其是动态内存分配和释放。该工具能够实时展示内存的分配和释放情况，帮助学生理解指针、引用和内存泄漏等问题。例如，在讲解动态数组的创建和销毁时，教师可以使用该工具展示内存的分配过程和释放过程。学生可以通过观察内存的变化，理解new和delete操作的正确使用方法，避免内存泄漏和野指针等问题。

（3）动态变量值展示。在程序运行过程中，变量的值会随着代码的执行而不断变化。通过该工具，教师可以在课堂上实时展示变量的值的变化，帮助学生理解变量的作用域和生命周期。例如，在讲解局部变量和全局变量的区别时，教师可以使用该工具展示变量在不同作用域中的值的变化。学生可以通过观察变量的值的变化，理解变量的作用域和生命周期，避免变量使用不当的问题。

表 3.2 传统 PPT 教学与去 PPT 式教学的对比

维度	传统 PPT 教学	去 PPT 式教学
知识点展示	静态展示，难以动态演示	动态展示，实时更新
学生参与度	被动接受，互动性差	主动参与，互动性强
理解难度	抽象概念难以理解	可视化展示，易于理解
实践支持	缺乏实时反馈	实时监控与反馈
个性化学习	难以满足个体差异	提供个性化学习路径

（4）项目实践中的应用。在项目实践中，该工具可以帮助学生更好地调试代码，发现和解决问题。例如，在开发学生成绩管理系统时，学生可以使用该工

具实时监控代码的运行情况，发现内存泄漏、指针错误等问题，并及时修复。通过该工具，学生可以在实践中不断优化代码，提高代码的质量和效率。同时，教师也可以通过工具的可视化数据，更好地指导学生，帮助他们理解和掌握复杂的编程概念。

去PPT式教学工具为语言程序设计课程的教学带来了新的思路和方法。通过可视化和动态分析的方式，学生能够更直观地理解复杂的编程概念，提高编程能力和问题解决能力，从而获取了更好的学习体验。

在传统的程序设计教学中，教师需要花费大量时间讲解知识点、审查学生代码和解答问题。而引入上述基于AI大语言模型的工具可以快速处理这些任务，为教师节省时间和精力，使教师能够将更多的时间用于引导学生思考和进行项目实践。同时，每个学生的学习能力和进度都不同，传统教学难以满足个性化的学习需求。AI 大模型可以根据学生的具体情况提供个性化的学习支持，使每个学生都能在适合自己的节奏和难度下学习，提高学习效果。所以，我们借助主流AI大语言模型（如Deepseek、豆包、kimi等），在课堂、实验和实践作业环节组织学生熟练掌握问题研判、思路分析、路线选择，效果验证的工作流程，例如采用VS Code扩展API开发插件实现代码语义分析等功能。在实际教学中，可以在如下方面提高学生的学习效率并巩固学习成果。

（1）理论讲解。教师在讲解 C++ 基础概念，如变量、数据类型、控制结构等时，可借助 AI 大模型提供更丰富的解释和示例。例如，利用知识问答系统讲授虚函数表实现机制问题，通过实时代码执行过程动画演示STL模板库中核心算法的流程等。

（2）知识点覆盖。利用 AI 大模型拓展教学内容，介绍 C++ 在不同领域的应用案例和最新发展趋势。比如，目前的教材使用的版本一般最高只支持到C++14，而AI大语言模型会在学生提问时，不仅仅给出C++98/03/11/14版本下传统的解决思路，还会介绍C++17/20/23等版本对这一问题的思考和改进的处理方式，从而有效弥补课程教学的缺陷和不足。

（3）创意启发。在进行 C++ 项目实践时，AI 大模型可以为学生提供项目创意和设计思路，并根据业界工程规范生成的UML图讲解设计模式，通过插件实时修改示例代码并展示多态效果。进而根据需求分析生成具体的代码框架或设计提示。学生可以在此基础上进行理解和修改，提高项目开发的效率。但教师也会注意引导学生深入理解代码的原理，避免过度依赖代码生成。

（4）代码审查与纠错。学生在编写 C++ 代码时，经常会遇到各种错误。AI大语言模型可以实现实时错误分析，并结合编译器输出，生成自然语言解释，同时结合教学内容，给出学生易懂的错误提示，从而巩固学生的知识点的掌握并增强印象。例如，当学生编写的函数调用出现参数不匹配问题时，AI 大模型能准确指出错误位置，并给出正确的调用方式。

（5）解决方案优化。对于学生编写的代码，AI 大模型可以分析其性能瓶颈，并提供优化建议。比如，在学生实现排序算法时，模型可以对比不同排序算法的复杂度，建议学生根据具体需求选择更合适的算法，提高代码的运行效率。

（6）个性化学习路径规划。根据学生的学习进度和能力水平，AI 大模型可以为每个学生制定个性化的学习路径。对于基础较弱的学生，模型可以推荐更多的基础练习和详细的知识点讲解；对于学有余力的学生，提供更具挑战性的项目和拓展知识。

通过上述举措，我们极大提高了课程教学的效率，拓宽学生的知识视野，增强学生的学习主动性，实现更高效的教学反馈与纠错，并真正做到因材施教。

4 教学效果评估

在程序设计教学实践中采取上述创新教学方法后，笔者发现学生对程序设计课程知识点的理解能力显著提高，在项目实践中能够更快地发现和解决代码中的问题，代码质量明显提高。通过可视化和动态分析的方式，学生对程序设计学习的兴趣明显提高，课堂参与度和自主学习的积极性显著增强。

笔者统计了 2024-2025 学年所在学院的《高级语言程序设计（C++）》课程的两轮完整教学实践数据。数据来源于：1）教务系统成绩库；2）教学平台（希冀+头歌）日志；3）AI 助教插件后台；4）课程结束后的匿名问卷四条证据链交叉验证“贯穿式案例 + AI 大语言模型”双轮驱动模式的成效，具体汇总如下。

4.1 学习成绩量化提升

为验证“贯穿式案例+AI大语言模型”双轮驱动模式的教学成效，本研究选取同年级两个平行班级（实验班采用新模式，对照班采用传统模式）进行对比分析。结果显示创新教学模式显著提升了学生的学业表现。

如图4.1.1所示，实验班期中平均分达 72.5分，较对照班（62.3分）提升 10.2分，期末平均分 69.1分，较对照班（58.7分）提升 10.4分，及格率（≥60分）从 58.3% 提升至 71.1%（+12.8%），优秀率（≥90分）从 4.4% 提升至 11.1%（+6.7%）。

依托“希冀”智能化教学平台，对实验班与对照班在程序设计课程（C++）中12项核心知识点的学习成效进行了严谨的对比评估，如图4.1.2所示。

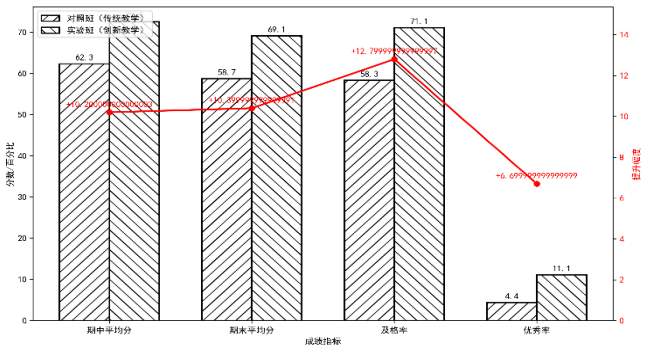

图4.1.1核心课程成绩对比

核心知识点涵盖从基础语法到高级抽象的关键编程概念，包括但不限于：变量与数据类型、运算符与表达式、控制结构、函数、数组与字符串、指针与引用、类与对象、构造函数与析构函数、运算符重载、继承、多态、模板等。通过平台采集的标准化后测数据分析，结果显示：实验班（采用特定的新型教学方法/干预措施，如项目驱动、精准推送、强化交互等）在知识掌握度上展现出全面且显著的优越性，其整体后测得分率显著高于采用传统教学模式的对照班，为相关教学改革和精准化教学策略的设计提供了有力的实证支持。

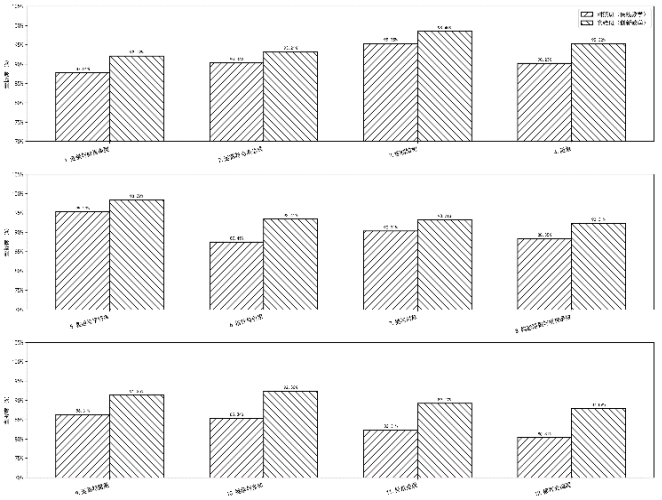

图4.1.2 知识点掌握度对比

实验班通过集成了LLM编程助手的编译环境（如Trae/Copilot + Visual Studio Code），提供实时代码诊断（语法/逻辑错误定位），个性化学习路径引擎（基于知识点掌握度动态推荐），同步C++最新版本标准特性扩展知识库，编程助手后台搜集的核心效益概要统计如图4.1.3所示。

4．2　实践能力提升指标

本课程以学生成绩管理系统作为课程大作业核心实践项目，要求实现多角色协作、支持动态扩展的C++应用系统。该项目设计贯彻工程教育认证（OBE）理念，从单纯编码能力评估转向全流程工程素养培养。通过对照班（传统教学）与实验班（AI赋能教学）的平行实施对比，如表4.2.1所示展示出三项提升。

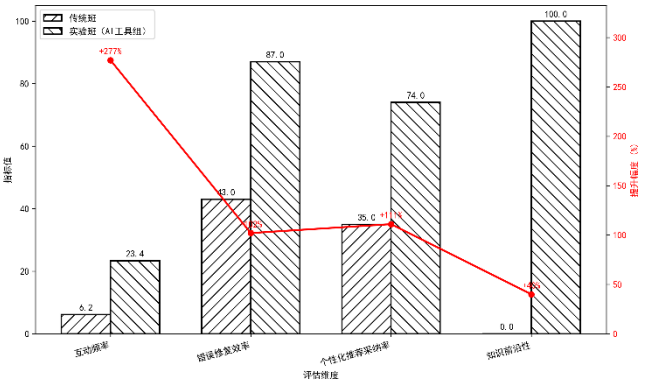

图4.1.3 AI工具使用效益对比表

（1）从功能实现到架构设计的认知跃迁。实验班的功能完整度不仅体现在基础CRUD操作，更展现出系统级设计思维，如采用MVC架构清晰分离业务逻辑（GradeService）、数据持久层（SQLiteAdapter）和交互界面，通过观察者模式实现成绩更新时的实时通知（如班主任邮箱预警），体现计算思维与领域知识融合。

（2）工程规范的内化养成。实验班代码规范达标率背后是工业化开发范式的落地，严格执行Git Flow工作流（特性分支→代码评审→CI集成），应用clang-format统一代码风格，静态分析消除潜在缺陷，文档驱动开发（注释率＞85%），提升系统可维护性。

（3）需求响应的敏捷重塑。当新增“成绩预测”需求时，实验班相应周期更短，代码平均修改成本更低，显著低于对照班。

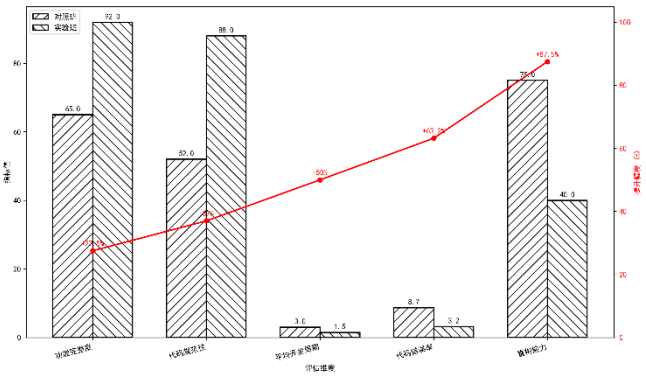

图 4.2.1 课程大作业效果对比

4．3　学习行为于态度转变

如图4.3.1所示，实验班在AI赋能的互动式教学环境下，学生的学习行为发生结构性转变，表现为三个维度的突破性提升：

（1）主动提问次数。高频深度提问表明学生从知识消费者转向解决方案设计者，契合教学目标中的分析-评价-创造高阶能力发展。

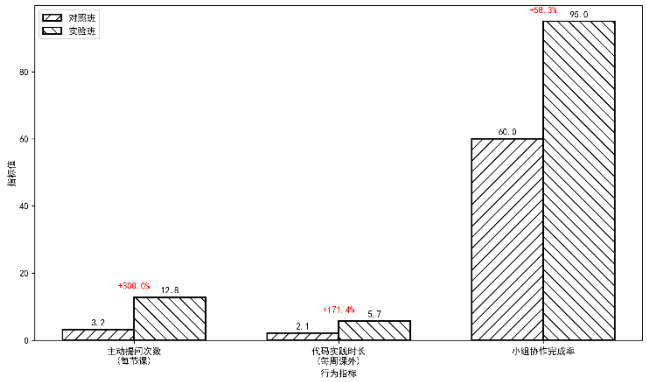

图 4.3.1 课堂参与度统计

（2）代码实践时长：从任务响应到心流沉浸，实验班周均课外实践5.7小时（对照班2.1小时）的背后是学习动机的重构：AI工具提供的即时正向反馈环，使编程实践从“苦役”转化为成瘾性创造活动。

（3）小组协作完成率：从形式分组到有机协同，完成率差距揭示协作模式的根本差异。行为数据的跃迁证明，AI赋能的课堂已突破传统教学的“三低困境”（低参与度、低持续性、低协作性）。实验班学生在即时反馈中迭代认知（提问）、通过游戏化机制保持专注（实践）、用技术中介实现无缝协作（小组）。这些为构建新一代智能教育范式提供了实证基础。

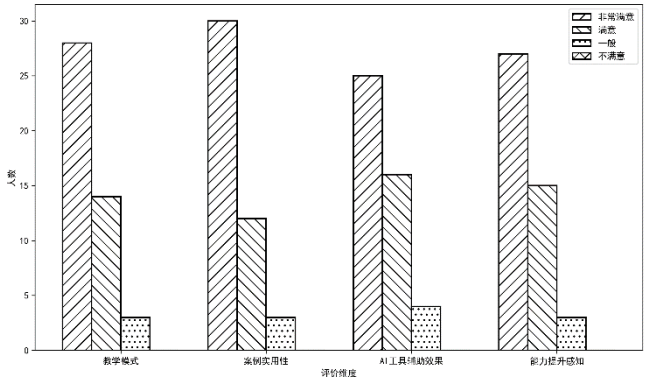

图 4.3.2 实验班学生满意度统计

通过问卷调查的反馈统计，如图4.3.2所示，实验班学生（采样人数45人）对AI赋能教学展现超高认同度，教学模式、案例实用性、AI工具辅助、能力提升感知这四个维度满意度均突破90%。尤其值得关注的是零不满意率与成绩靠后（30%）学生中b68%表示“非常满意” 的叠加现象，印证该模式成功突破传统编程教育的三重困境：

（1）通过实时AI纠错化解挫败感；

（2）借力产业级案例弥合学术/产业鸿沟；

（3）依托动态路径推荐使薄弱组任务完成率。四大维度满意度构成教育数字化转型的有效指标，标志“精准反馈-实践驱动-能力可视”的新范式已获得学生深度共鸣。

5 结论与展望

高级语言程序设计课程教学与AI技术的融合将具有更广阔的发展前景。随着 AI 技术的不断进步，AI 大模型的性能和准确性将不断提高，其在高级程序设计语言教学中的应用也将更加深入和广泛。例如，未来的主流AI大语言模型可能能够实现更加精准的个性化学习路径推荐，根据学生的实时学习情况和反馈，动态调整学习计划和内容，真正实现因材施教。

在教学实践方面，我们可以进一步拓展教学实践案例的类型和难度，引入更多实际工程项目，让学生在更复杂的场景中锻炼编程能力和解决问题的能力。同时，加强对学生计算思维和创新能力的培养，鼓励学生在项目中尝试新的技术和方法，提高学生的综合素质和竞争力。

未来的研究可以关注如何更好地整合 AI 技术与高级程序设计语言教学资源，开发更加智能化的教学工具和平台。例如，开发基于 AI 的编程学习辅助工具，能够实时监测学生的编程过程，提供即时的错误提示和优化建议；构建智能化的教学资源库，根据学生的学习需求和进度，自动推荐合适的教学视频、练习题和项目案例等。此外，还可以深入研究 AI 技术对学生学习心理和学习行为的影响，为教学方法的改进提供更科学的依据。

参 考 文 献

[1] 李国和、董丹丹. 面向胜任力培养的程序设计综合实践教学探索[J]. 计算机技术与教育学报, 2023, 11(11): P127-132

[2] 张策、张小东. 新工科背景下“计算机程序设计基础”课程教学改革探索[J]. 计算机技术与教育学报, 2024, 7(12): P85-90

[3] Li X, Wang Y, Zhang H. A Hybrid Teaching Model Combining Project-Based Learning and AI-Assisted Coding for Programming Courses[J]. IEEE Transactions on Education, 2023, 66(2): 189-198.

[4] 王宏志，李建中。计算机科学与技术学科课程体系与课程建设研究 [J]. 中国大学教学，2021 (3): 34-39.

[5] 蒋社想. 深度学习视角下高级语言程序设计混合式课程教学方法探索[J]. 计算机技术与教育学报, 2024, 8(12): P160-164

[6] Stroustrup B. Evolution of C++: 1979-2020[C]//Proceedings of the ACM on Programming Languages. 2020, 4(OOPSLA): 1-23.

[7] 谢晓艳、谢晓巍、曹伟. 面向能力培养的程序设计基础课程改革实践[J]. 计算机技术与教育学报, 2022, 9(10): P90-93

[8] 李志刚、杨吉斌、张睿、王彩玲、陈卫卫. 基于ChatGPT 的程序设计翻转课堂教学方法实践[J]. 计算机技术与教育学报, 2023, 08(11): P125-129

[9] 教育部高等学校计算机类专业教学指导委员会。高等学校计算机类专业发展战略研究报告暨核心课程体系[M]. 北京：高等教育出版社，2022.

[10] 罗珣、胡学钢、方宝富、李建华. 基于兴趣度-持久度的程序设计课程教学改革探讨[J]. 计算机技术与教育学报, 2021, 11(9): P80-85

[11] 林健. 面向未来的中国新工科建设[J]. 清华大学教育研究, 2017, 38(2): 26-35.

[12] 韩冰,郭咏梅,候慧玲.以兴趣为导向“面向对象程序设计 C++”教学实践[J].软件工程,2016,19(6):59-60.