

人工智能代码生成在计算机学科编程教学中的利弊探讨

邵苏杰, 陈兴渝

北京邮电大学计算机学院, 北京

收稿日期: 2026年9月3日; 录用日期: 2026年10月6日; 发布日期: 2026年10月12日

摘要

以大语言模型为基础的人工智能(Artificial Intelligence, AI)代码生成工具正在飞速进入C语言程序设计、Java语言程序设计、数据结构与算法等计算机学科核心编程课程的教学过程中。学生可以通过自然语言描述快速获得代码、错误解释与修改建议, 教师也可以利用相关工具生成示例、练习和反馈。人工智能代码生成由此极大降低了编程学习和实践的操作门槛, 但同时也模糊了完成代码与掌握编程之间的区别, 给计算机学科编程教学带来了明显的冲击和变化。本文结合计算机学科编程教学特点及作者近几年课程教学过程和成绩资料, 对人工智能代码生成的教学收益和潜在风险等利弊进行讨论。作者讲授的北京邮电大学2022~2026年计算机相关编程学科成绩显示, 在平时作业可接触人工智能、期末考试禁用人工智能的教学情境中, 平时成绩与期末成绩存在明显落差。这一现象不能直接证明人工智能造成了学生能力下降, 却提示以最终代码为主的平时评价中人工智能并没有完全助力大部分学生良好地掌握独立编程能力。本文认为, 人工智能代码生成的主要价值在于提供及时示例、解释和反馈, 节省一些不必要的简单重复过程以提高效率, 主要风险则在于学生把问题分析、算法设计、代码阅读和调试等认知活动一并交给工具, 实际动手能力练习培养某种程度上有所下降, 利弊共存。因此, 计算机学科编程教学中不宜简单禁止或放任人工智能, 而应把它限定为可质疑、可验证的辅助工具, 但同时保留必要的独立思考和脱离人工智能完成任务的机会。

关键词

人工智能代码生成, 计算机学科, 编程课程, 独立编程能力

Discussion on the Advantages and Disadvantages of AI Code Generation in Computer Programming Teaching

Sujie Shao, Xingyu Chen

Abstract

Artificial Intelligence (AI) code generation tools based on Large Language Model are rapidly entering the teaching process of core programming courses in computer science, such as C programming, Java programming, data structures and algorithms. Students can quickly obtain code, error explanations, and modification suggestions through natural language descriptions, while teachers can also use related tools to generate examples, exercises, and feedback. AI code generation has thus greatly lowered the operational threshold for programming learning and practice. However, it has also blurred the distinction between completing code and mastering programming, bringing significant impacts and changes to computer science programming teaching. This paper discusses the advantages and disadvantages of AI code generation, including its teaching benefits and potential risks, based on the characteristics of computer science programming teaching and the author's teaching process and examination data in recent years. The author's teaching examination data for related computer programming courses at Beijing University of Posts and Telecommunications from 2022 to 2026 shows a significant gap between regular performance and final examination scores in a teaching scenario where AI is accessible during regular assignments but prohibited during final exams. This phenomenon does not directly prove that AI has caused a decline in student abilities, but it suggests that AI does not fully assist most students in mastering independent programming skills in regular evaluations based primarily on final code. This paper argues that the main value of AI code generation lies in providing timely examples, explanations, and feedback, saving some unnecessary simple repetitive processes to improve efficiency. The main risk lies in students handing over cognitive activities such as problem analysis, algorithm design, code reading, and debugging to tools, which to some extent reduces the cultivation of practical skills. Therefore, in computer science programming teaching, it is not appropriate to simply prohibit or allow AI unchecked. Instead, it should be limited to a questionable and verifiable auxiliary tool, while retaining the opportunity for necessary independent thinking and completing tasks without AI.

Keywords

AI Code Generation, Computer Science, Programming Courses, Independent Programming Ability

Copyright © 2026 by author(s) and Hans Publishers Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

1. 引言

人工智能(Artificial Intelligence, AI)技术正在快速进入教育领域, 利用 AI 代码生成工具快速生成、解释和修改程序代码逐渐成为计算机学科编程课程学习中的常见现象。学生可以借助 AI 查询语法、分析报错、得到修改建议、获得示例乃至完成整段程序, 作业完成效率大幅提升, AI 的学习支持能力展现无疑。教师也可以利用相关工具生成练习、测试数据和初步反馈, 从而提高教学准备的效率[1]-[3]。但程序设计编程课程的目标并不只是获得可以运行的代码, 还包括问题分析、算法设计、代码阅读、测试调试和实际应用等能力; 当学生把这些关键过程一并交给 AI 时, 作业完成效率的提高并不意味着独立编程能力的

同步提升, 教学过程中平时评价也可能难以区分学生的真实理解程度与工具辅助程度[4]-[6]。AI 代码生成在计算机学科编程教学中利弊并存, 需进一步讨论探索。

本文结合作者近几年讲授的北京邮电大学 2022~2026 年 C 语言程序设计、Java 语言程序设计、数据结构与算法等课程的成绩资料, 对平时可以接触 AI 而期末考试禁止使用 AI 的教学情境下的 AI 使用利弊进行分析。成绩资料呈现出平时成绩较高而期末成绩相对较低的现象, 但由于数据未记录学生个体使用 AI 的频率和方式, 且考试难度、学生基础与学习投入等因素也可能影响成绩, 本文不据此认定 AI 导致了学生独立编程学习掌握能力下降, 而是将其作为任务完成表现与独立编程表现可能存在差异的现实信号。在此基础上, 本文主要讨论 AI 代码生成对学生学习和教师教学的有利影响、过度依赖可能带来的能力弱化风险, 以及 AI 作为辅助工具进入计算机编程教学时应当保持的基本边界和应用思路。

2. AI 代码生成在计算机学科编程教学中的主要用途

(1) 学生端: 代码生成与算法比较

学生使用 AI 的方式具有明显层次差异。较低程度的使用包括查询语法、分析编译错误、寻找函数示例、解释陌生代码和直接生成代码, 这些行为与查阅教材、搜索技术文档较为接近, AI 辅助学生提升学习效率。更进一步的使用是让 AI 比较不同算法、补充测试用例或评价代码可读性, 此时工具承担了对话式辅导者的部分功能, AI 辅助学生提升对编程的理解。但占比较大且风险最高的方式则是把完整题目直接输入模型, 复制输出结果后稍作格式调整便提交, AI 仅辅助任务完成, 学生几乎无任何提升。三种方式使用的是同一种技术, 却对应完全不同的学习投入和效果。

国内关于程序设计教学的研究已经开始关注这一变化。相关研究指出, 生成式 AI 能够进入计算机编程课程的知识学习、实践操作和个性化辅导环节[7], 并可在智能协同教学、个性化学习、真实实践场景以及实时反馈等方面发挥作用[8]。这意味着学生接触 AI 并非偶然现象, 而是编程学习环境发生的现实变化。

(2) 教师端: 材料生成与反馈辅助

教师使用 AI 的价值主要体现在重复性材料生成工作上。编程课程需要大量例题、练习、测试数据和错误样例, 同一知识点还需要针对不同基础的学生提供不同解释。已有研究对习题生成、自动解题和辅助批阅等用途进行了讨论, 认为生成式人工智能能够为教师教学和学业辅导提供支持[3]。显然, AI 更适合充当材料初稿和辅助意见的生成者, 还不能自动替代教师决定教什么、如何教以及何为正确。

教学反馈是另一项 AI 辅助教师教学的适用功能。在大班教学中, 教师难以及时逐一阅读所有代码, AI 可以先标记语法、概念和复杂度方面的疑点, 再由教师处理共性问题和关键误解。相关研究将教学评一体化的实时反馈与评估视为生成式人工智能进入程序设计课程的重要应用方向[9]。这里的收益并不来自 AI 比教师更懂教学, 而是来自 AI 扩大了反馈覆盖面并缩短了等待时间。

(3) AI 代码生成用途收益

AI 代码生成工具能否产生学习收益, 取决于它替代了什么。如果它替代的是重复查找、机械输入或等待反馈的时间, 学生可以把精力投入算法比较和错误分析, 工具具有明显的辅助价值。如果它替代的是理解题意、构造方案和检查结果, 效率提升便可能以学习过程缩水为代价。将逆向工程思维引入生成式人工智能支持的编程学习, 能够突出学习者对生成结果进行分析、理解和再加工的重要性[10]。这从侧面说明, AI 使用次数增加或代码完成速度提高, 并不能自动转换为更高水平的程序设计能力。

3. AI 代码生成对计算机学科编程教学的有利影响

(1) 缩短获得示例和反馈的时间

初学者经常因一个标点、符号、变量类型或循环边界错误而长时间停滞。传统课堂中, 教师无法随

时回应每名学生, 搜索引擎返回的信息又常常超出学生的理解范围。AI 可以依据当前代码和报错信息给出针对性解释, 使学生迅速恢复练习, 减少无效等待, 也有助于维持学习热情。尤其在语法入门阶段, 学生若能及时知道错误出现在哪里、修改会产生什么结果, 便更容易建立代码与运行行为之间的联系。

AI 还可以根据学生追问改变解释粒度。例如, 同一段递归程序既可以用调用栈说明, 也可以用逐步执行表格说明; 同一排序问题可以给出稳定算法和不稳定算法。教师面对大班学生时难以持续提供如此多的个性化表达, AI 却可以快速生成多个解释候选。只要学生把这些解释当作需要验证的材料, 而不仅是答案, 工具就能扩大学习资源的可获得性和更多的学习辅助作用。

(2) 降低初学者的实际操作门槛

编程学习包含概念理解和实际操作两个方面。部分学生并非不理解问题, 而是受困于开发环境、函数名称和语法细节, 迟迟无法将自己的算法设计编程为可运行程序。AI 可以补充模板、演示 API 用法、转换不同编程语言, 从而降低操作性障碍。学生先看到一个可运行版本, 再逐步追问其结构, 有时比长时间面对空白编辑器更容易取得积极的学习效果。

这种便利对于跨专业课程和基础差异较大的班级更为明显。非计算机专业学生可能不以成为精英软件开发者为目标, 但仍需理解数据处理、算法和编程思维。适度的代码生成可以使其较快获得结果, 并把注意力转向问题含义。

以程序设计能力培养为导向的个性化教学研究表明, AI 在识别学习差异、提供针对性资源和支持实践训练方面具有应用空间[11]。这一价值的成立仍以能力培养为前提, 不只以作业完成率或代码生成速度评价效果。

(3) 支持多方案比较和反思

同一道题往往存在多种实现方式。AI 能够快速给出不同结构的代码, 为复杂度、可读性和健壮性比较提供材料。教师可以让学生判断哪一种方案更符合当前课程要求, 找出隐藏的边界条件, 并说明为何保留或拒绝某段生成结果。这类活动把 AI 输出转化为分析对象, 而不是提交答案。

从专业实践看, 阅读、审查和维护他人代码本来就是程序员的重要工作, 在 AI 大范围应用的情况下, 可能更加重要。让学生分析 AI 代码, 既能训练代码理解, 也能培养对自动生成结果负责的意识。AI 在此处不负责完成学习, 而是持续提供可供质疑的样本。

(4) 提高教师备课和辅助答疑效率

教师可利用 AI 快速生成练习、边界测试、常见错误版本和课堂演示素材。同一知识点可以按照不同难度设置题目, 也可以让工具把抽象概念转化为贴近专业情境的案例。对于学生频繁提出的基础问题, AI 能够提供第一轮解释, 教师则集中处理概念误解和复杂问题。

这种分工不会削弱教师作用, 反而要求教师承担更高层次的筛选与判断。生成内容是否符合教学进度、是否采用了尚未讲授的语法、是否存在安全或效率问题, 均需教师决定。AI 节省的是初稿制作和重复表达时间, 不能替代课程目标设计和专业把关。

4. AI 代码生成可能带来的不利影响

(1) 问题分析和算法设计被认知外包

本文所说的认知外包是指学生把原本需要自己完成的问题分析、方案设计和结果判断交给 AI 工具, 从而降低自身在任务中的认知投入。有关教育生态影响的研究指出, 生成式人工智能可能带来知识异化、学生主体性异化和教学过程异化等风险[6]。在程序设计编程中, 合理借助 AI 工具查询语法并无明显问题; 但若把题意分析、数据结构选择、算法构造和正确性判断全部交给 AI, 学生完成的便主要是提示输入和结果搬运, 而不是编程问题求解。

程序设计能力具有过程性。学生需要经历理解需求、提出方案、发现错误和修正方案的循环,才能形成可迁移的知识。AI 直接给出的完整代码压缩了这一过程。代码越快出现,学生越容易产生已经会了的感觉,却未必能够在条件变化时重建方案。认知负担下降本身不是坏事,关键在于被减掉的是机械负担还是具有学习价值的思考。

(2) 代码阅读、修改与调试能力可能弱化

生成代码并非天然易于使用。初学者缺少知识基础,往往难以识别看起来合理的错误。当程序无法运行时,他们可能继续要求 AI 重写,而不是根据报错、变量状态和测试结果定位原因。需要组织人机协作编程学习,能够促使学习者对生成结果进行拆解、追溯和重新建构,而不是停留在接受结果的层面[10]。

这种工作方式会影响修改能力。真实问题通常包含模糊需求、已有代码约束和不断变化的条件,程序员不能每次都从头生成完整的正确程序。学生如果只熟悉描述需求-复制答案的路径,就难以在局部修改时判断哪些变量、函数和模块会受到影响。长期来看,代码所有权与责任感也会被削弱:学生提交了程序,却不知道为何这样写,更无法保证程序在新数据上的行为。

(3) 输出可能正确,也可能只是通过了有限测试

AI 生成代码具有语言形式上的流畅性,却不保证功能正确。相关研究指出,生成式人工智能可能产生事实错误和不可靠内容,教育使用者不能因表达完整便直接接受其输出[11]。这一问题在程序代码中表现为:编译成功、通过示例测试和满足完整需求并不是同一件事。

初学者往往用教师提供的少量样例判断程序是否正确,AI 也容易生成恰好适配这些样例的代码。边界值、空输入、异常路径、性能限制和安全问题可能被忽略。如果学生没有测试意识,生成工具反而会放大表面正确性。工具越能写出规范、完整的代码,使用者越需要主动寻找反例。

(4) 平时评价可能出现失真

以最终代码和运行结果为主的作业评价,难以区分学生独立完成、同学协助和 AI 生成。高分可能表示程序确实运行正确,却不能完整表示学生理解了问题。当 AI 可以快速生成基础题答案时,传统作业的区分度会下降,平时成绩逐渐混合了编程能力、工具使用能力和外部帮助程度。

有关学习评价的研究指出,生成式人工智能能够直接参与学习成果生产,对以最终答案为中心的评价方式造成冲击[4]。就成绩解释而言,问题不只是学生是否不合理使用,还在于现有任务是否仍能测量预期能力。若平时作业主要测量获得正确编程结果的能力,而期末考试测量脱离 AI 工具后的独立编程能力,两类成绩自然可能出现较大差异。

5. 平时成绩与期末成绩差异的现实观察

本文对作者讲授的北京邮电大学 2022~2026 年 C 语言程序设计、Java 语言程序设计、数据结构与算法等课程的 9 份成绩文件进行了匿名汇总,共 427 条可配对的平时与期末成绩记录。平时成绩主要来自作业、练习或过程性考核,相关教学活动允许学生接触 AI。期末考试成绩则来自不允许使用 AI 的 2 小时左右集中机试。

Table 1. Comparison of regular performance and final examination scores

表 1. 平时和期末成绩对比表

指标	平时成绩	期末成绩
平均分	94.28	47.09
中位数	100	45
及格率	95.78%	40.05%

表 1 的汇总分析结果显示,平时成绩平均比期末成绩高 47.19 分,期末中位数成绩达不到及格,及格率相差超过 55%,大部分学生平时能够完成作业,给出正确的编程结果,期末集中机试却很难完成可执行的符合考核要求的编程设计。

Table 2. Gap between regular performance and final examination scores

表 2. 平时和期末成绩差异表

指标	数值
平时成绩高于期末成绩的学生占比	89.46%
平时成绩高于期末成绩 20 分的学生占比	78.22%
平时及格但期末不及格的学生占比	58.19%

表 2 显示 89.46%的学生期末成绩低于平时成绩,78.22%的学生差距达到 20 分及以上。在平时成绩及格的学生中,58.19%在期末考试中未达到 60 分。不同课程、年份和班级的具体差距并不完全相同,但平时分集中在高分段、期末分明显分散并整体偏低的现象具有较强一致性。

(1) 成绩落差能够说明什么

这组数据首先说明,平时成绩和期末成绩测量的形式和内容不同,以及这种不同可能带来的结果不同。平时作业允许查阅资料、反复修改并使用 AI,反映的是在时间充足及外部支持下完成任务的表现。期末考试限制工具并没有时间压力,更接近学生独立调用知识的能力。二者本来不必完全一致,存在成绩差异是正常的。平时高分的获得可能有多种方式,可能是学生使用了 AI 工具后直接获取正确代码后获得,也可能是学生使用 AI 工具获取正确代码并对其深入理解掌握后获得,也可能是学生未使用 AI 工具且通过对教师讲授内容的理解掌握自主编写调试完成正确代码后获得,也可能是学生在其他同学的指导协助下获得的。期末考试高分要求在没有 AI 工具辅助的情况下在短短 2 小时内完成数道编程题目的正确的可执行程序,语法和算法掌握乃至编程习惯、调试方法等都需要扎实掌握,获得难度较大。说明学生的成绩受学习环境、考核环境、外部工具等多种因素的影响。但如此明显的差距提示,平时高分不能直接解释为学生已经形成同等水平的独立编程能力。

成绩落差也与前述认知外包风险相符。部分学生可能利用 AI 跨越了代码生成环节,却没有同步形成解释、调试和迁移能力。当 AI 外部支持在期末考试中被撤除,平时作业中被工具承担的认知活动重新落到学生身上,能力缺口便会显现。成为一种成绩落差大的可能情形,在某种意义上增加了成绩落差大的概率。

(2) 成绩落差不能证明什么

成绩数据不能单独证明 AI 导致了期末低分。首先,成绩表没有记录每名使用 AI 的频率、用途和依赖程度,无法建立 AI 使用量 - 能力变化的个体关系。其次,不同课程和年份的考试难度、教师要求、学生基础及计分方式存在差异。再次,期末考试还受到时间压力、复习投入和应试状态影响。把所有差距归因于 AI,会把相关关系误写为因果关系。

因此,本文将成绩现象作为教学问题的信号,而不是因果证据。更准确的结论是:在平时可以获得 AI 支持、期末必须独立完成的情境中,学生的任务完成表现与独立编程表现出现明显失配;AI 依赖可能参与了这一过程,但其作用程度仍需结合匿名问卷、访谈、作业过程记录或对照研究进一步判断。现有国内程序设计教学研究多肯定 AI 在个性化学习、过程反馈和实践支持方面的价值[7][10],但这些价值是否能够稳定转化为脱离工具后的独立能力,仍需要更有针对性的实证研究。

6. 对 AI 辅助计算机学科编程教学的几点认识

(1) AI 适合承担提示和反馈,不宜替代核心思考

AI 的合理角色可以概括为帮助学生继续思考, 而不是替学生结束思考。当学生遇到语法错误、概念困惑或测试失败时, 工具可以提供线索、反例和解释。当任务要求分析需求、选择算法和判断正确性时, 学生仍应保留主要认知责任。判断一次使用是否有利于学习, 可以追问: 工具给出答案后, 学生是否更理解问题, 是否能够说明选择依据, 是否能够在条件变化时修改代码。如果答案是否定的, 即使程序运行成功, 辅助作用也很有限。

(2) 评价 AI 代码比复制 AI 代码更有学习价值

教师无需把 AI 完全挡在课堂之外, 可以把生成结果转化为教学材料。让学生比较两段 AI 代码、寻找设计差异、补充测试用例、解释复杂度或按照新需求修改程序, 能够把工具优势与代码审查能力结合起来。AI 输出的不确定性在这种活动中不再只是风险, 也成为训练批判性判断的资源。

这一思路同样适用于学生日常提问。与其直接要求写出完整代码, 更有学习价值的提问是指出当前代码可能出错的位置、给出一个反例但不要给出完整答案、解释为什么选择相应的数据结构和算法、问题变化后解释为什么修改等。提示方式不能保证学习, 但能决定工具更接近答案机还是辅导者。

(3) 必须保留独立完成任务的机会

使用 AI 与独立编程并不矛盾。课程可以在部分练习中允许充分使用工具, 同时保留无需 AI 完成的基础任务、现场解释和调试活动。独立练习的目的不是模拟没有 AI 的未来工作环境, 而是确认学生具备判断 AI 结果所需的基础能力。没有代码阅读、算法设计和测试基础, 学生就难以对生成结果负责。

期末考试禁用 AI 能够提供一种独立能力观察, 但它不是唯一方式。课堂提问、代码走查、局部修改和现场调试也能了解学生是否真正理解。本文并不主张立即改变考核结构, 只强调平时成绩的解释需要结合任务条件: 允许 AI 完成的作品可以反映人机协作表现, 不能自动等同于脱离 AI 的独立能力。

(4) 应适当增加对生成内容的专业判断教学

AI 可以提高材料生成和反馈效率, 却不能承担最终教学责任。教师也需要检查示例是否符合课程进度、代码是否正确、解释是否准确以及任务是否真正指向学习目标。将大语言模型融入程序设计课程, 仍需把教师组织、过程设计和以学生为中心作为教学实施的重要条件[8]。因此, 教师需要把 AI 输出作为候选材料, 而不是未经审核的标准答案。

那么将教师使用 AI 准备教学的过程和内容适当向学生公开, 可能是一种潜在的更好的教学方式。当学生看到教师在验证、修改和拒绝 AI 建议时, 更容易理解专业能力并非熟练复制结果, 而是对结果做出判断, 也能更容易地接受教师传授的程序设计编程整体思维。AI 素养的核心不是提示词技巧和结果, 而是知道工具能做什么、不能保证什么, 以及如何让工具辅助自身能力提升。

(5) 避免把讨论简化为禁止或全面开放

禁止 AI 会忽略其巨大的技术和应用价值。但全面开放则可能使基础薄弱的学生在尚未形成判断能力之前产生依赖。两种极端立场都把复杂教学问题变成了工具态度问题。真正需要区分的是使用目的和认知责任。查找解决语法错误与生成整份作业不同, 比较方案与直接提交不同, 能够解释和修改的使用与无法说明代码含义的使用也不同。

AI 代码生成的计算机学科编程教育价值并不由技术能力单独决定, 而由教学情境、任务要求和学生参与方式共同决定。生成式人工智能的教育应用最终仍应指向人的发展和学生主体性的实现[12]。在计算机学科编程教学中, 这一原则可以具体化为: 工具可以生成代码, 但理解和验证仍属于学生与教师, AI 可以使用, 但需针对性有目的有边界的使用。

7. 结语

AI 代码生成已经改变计算机学科编程课程中的作业完成方式和教学资源生产方式。它能够为学生提

供即时示例、错误解释和多种方案,也能帮助教师生成练习、测试和反馈。对于基础薄弱或需要及时帮助的学生,这些功能具有现实价值。若使用得当,AI可以减少机械性障碍,使学习者把更多精力投入比较、分析和改进。

同样,风险也很清楚。学生可能把原本应由自己完成的问题分解、算法设计、代码阅读和调试过程交给工具,从而形成高任务完成度与低独立能力并存的状态。作者近几年讲授的课程成绩呈现的平时高、期末低现象与这一风险相符。虽然不能被当作AI造成能力退化的直接证明,但也是一种提醒:在AI可用环境中获得的成绩,需要谨慎解释其能力含义。

AI代码生成进入计算机学科编程教学后,关键问题已不再是学生能否获得代码,而是学生能否理解、验证、修改并对代码负责。教师也不必在全面禁止和无条件接受之间二选一。把AI限定为提示、解释、比较和反馈工具,同时保留独立实践,是当前条件下较为稳妥的认识。技术可以减少编写代码的部分成本,但教育仍需守住分析问题、理解问题、设计方案、动手实践这一核心学习历程。

参考文献

- [1] 朱永新, 杨帆. ChatGPT/生成式人工智能与教育创新: 机遇、挑战以及未来[J]. 华东师范大学学报(教育科学版), 2023, 41(7): 1-14.
- [2] 杨宗凯, 王俊, 吴砥, 陈旭. ChatGPT/生成式人工智能对教育的影响探析及应对策略[J]. 华东师范大学学报(教育科学版), 2023, 41(7): 26-35.
- [3] 卢宇, 余京蕾, 陈鹏鹤, 李沐云. 生成式人工智能的教育应用与展望——以 ChatGPT 系统为例[J]. 中国远程教育, 2023, 43(4): 24-31, 51.
- [4] 沈书生, 祝智庭. ChatGPT 类产品: 内在机制及其对学习评价的影响[J]. 中国远程教育, 2023, 43(4): 8-15.
- [5] 王佑镁, 王旦, 梁炜怡, 柳晨晨. “阿拉丁神灯”还是“潘多拉魔盒”: ChatGPT 教育应用的潜能与风险[J]. 现代远程教育研究, 2023, 35(2): 48-56.
- [6] 周洪宇, 李宇阳. ChatGPT 对教育生态的冲击及应对策略[J]. 新疆师范大学学报(哲学社会科学版), 2023, 44(4): 134-144.
- [7] 张红卓, 周小宝, 许玉焕, 杨婧. 生成式人工智能赋能计算机程序设计类课程教学创新[J]. 计算机教育, 2024(7): 44-48.
- [8] 高洪皓, 陈章进. 人工智能赋能程序设计课程教学改革[J]. 计算机教育, 2024(7): 41-43, 48.
- [9] 赵璐, 蔡惠, 王烟濛, 周剑. 生成式人工智能赋能程序设计课程教学研究[J]. 大学教育, 2025(8): 33-39.
- [10] 翟雪松, 张丽洁, 夏亮亮, 徐鑫, 朱强. 基于 GAI 的逆向工程教学思维在人机协作中的应用研究——以编程教育为例[J]. 电化教育研究, 2024, 45(9): 61-68.
- [11] 苏小红, 苗启广, 陈文字. 基于 AI 赋能和产教融合提升程序设计能力的个性教学模式[J]. 中国大学教学, 2023(6): 4-9.
- [12] 周玲, 王烽. 生成式人工智能的教育启示: 让每个人成为他自己[J]. 中国电化教育, 2023(5): 9-14.