大模型时代编程基础课程的重构路径探索*

张晓莹　孙荟哲　季燎原　祁鲁

中国计量大学现代科技学院，义乌 322000

摘　要　随着智能编程工具广泛应用，编程语言教学面临如何适应新环境的挑战。研究构建知识图谱驱动的课程重构框架，通过对比人工编写代码与大模型生成代码时的认知差异，提出“基础保留 - 思维强化 - 技术整合”三阶段教学内容筛选机制，并开发“人工 - 生成”双轨教学资源。实验数据显示，该教学模式能有效提升学生算法设计能力（平均提高 11.2 分）和计算思维抽象能力（平均提升 13 分）。研究给出平衡技术辅助与核心能力培养方法，为编程教育变革提供完整解决方案。

关键字　大语言模型，编程教育重构，知识图谱，计算思维

Exploring the Restructuring Path of Programming Foundation Courses in the Era of Large Models *

Zhang Xiaoying　Sun Huizhe　Ji Liaoyuan　Qi Lu

China Jiliang University College of Modern Science and Technology, Yiwu 322000, China;

Abstract—With the widespread application of intelligent programming tools, programming language education faces the challenge of adapting to the new environment. This research constructs a knowledge graph-driven curriculum restructuring framework. By comparing the cognitive differences between manually written code and code generated by large models, a three-stage teaching content selection mechanism—"foundation retention, thinking enhancement, and technology integration"—is proposed, along with the development of dual-track teaching resources for "manual and generated" code. Experimental data show that this teaching model effectively improves students' algorithm design skills (with an average increase of 11.2 points) and computational thinking abstraction abilities (with an average improvement of 13 points). The study provides a method to balance technical assistance with the cultivation of core competencies, offering a comprehensive solution for the transformation of programming education.

Keywords—Large language models,Programming Education Restructuring,Knowledge Graph,Computational Thinking

1　引　言

1.1　研究背景与问题提出

大模型快速发展深刻改变编程教育格局[1]，以 GPT 系列为代表的大语言模型能依据自然语言描述自动编写代码[2]，且在编程课程评测中达到人类相当表现[3]，对传统编程教学模式产生颠覆性影响，尤其对独立学院计算机专业学生的学习方式带来转型挑战。

传统教学注重语法规则学习，通过机械记忆和重复训练标准范例实现教学效果，这种模式在大模型时代暴露出结构性矛盾[4]。调查显示初学者优先使用智能编程助手完成作业，导致其底层语法理解深度和算法设计能力下滑[5]，且大模型生成代码在风格统一性和教学适配性上存在缺陷，可能造成知识断层。

代码自动生成虽然降低了编程入门门槛，但弱化了计算思维的系统性培养（如图 1 所示），该矛盾在独立学院更突出——学生学习基础参差不齐、自主学习能力较弱。如何重构编程基础课程体系，在发挥大模型优势的同时保障计算思维培养，成为计算机基础教育改革关键问题。

1.2　研究现状与关键挑战

国内外学者围绕大模型对编程教育的影响开展研究，高等教育阶段利用大模型实现编程练习自动评分[6]、错误检测[7]等辅助教学，基础教育阶段尝试将代码生成工具整合到 Python 入门课程并限制使用范围以平衡效率与学习效果，教学实践层面探索大模型在编程课堂的应用路径[8]和创新模式[9]。这些研究为技术赋能教育提供参考，但课程体系重构层面存在显著空白。

当前研究面临三大挑战：一是突破传统“语法掌握-算法实现”线性培养模式，建立适应智能编程环境的新型能力框架；二是缺乏系统方法论判定编程知识的强化方向与技术替代边界；三是现有评价标准以代码正确性为主，难以评估计算思维发展水平。学生过

*基金资助：浙江省十四五教学改革项目《人工智能赋能程序设计课程的“学-用-创-辨”教学模式研究》，编号：JGBA2024786 。

早依赖代码生成工具会导致知识碎片化、思维能力空心化。

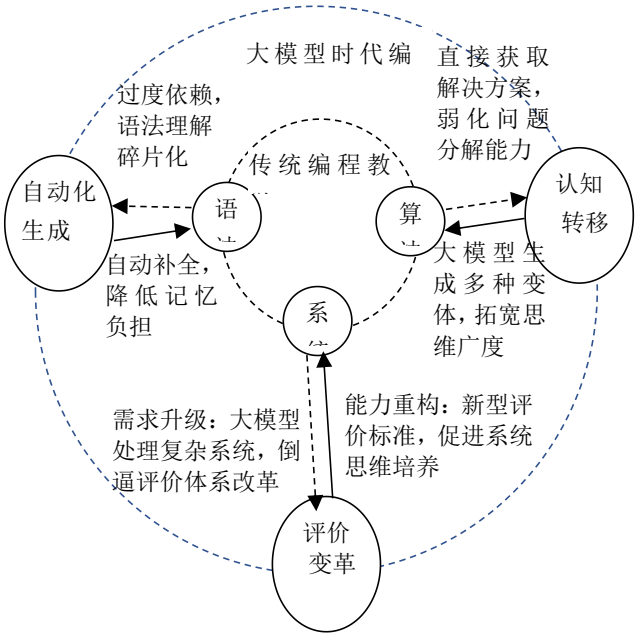

图 1 大模型对编程教育的影响机制

1.3 研究框架与方法

本研究提出知识图谱驱动的课程重构方法论，构建认知分析、资源融合、思维培养三维研究框架。

认知分析采用对比实验法，考察大模型生成代码与人工编写代码在认知负荷、学习迁移效果的差异，为课程目标制定提供实证依据。

资源融合通过构建“人工范例-大模型生成案例”双轨资源库实现传统教学与技术创新的结合，实施三阶段生产流程：教师团队编写标准案例、基于相同需求生成大模型解决方案、专家标注形成含思维对比注解的教学资源包。

2 大模型代码生成特征分析

2.1 语法结构与逻辑范式比较

大模型生成代码与人工编写代码在语法结构和逻辑范式上存在系统性差异。典型 C 语言案例对比显示，大模型生成代码控制结构呈模式化特征——处理循环时倾向标准化 for 循环模板，而人工代码会根据场景灵活选择 while 或 do-while 结构，该差异反映机器对确定性模式的依赖，与人类编程的情境化决策本质不同。

变量命名方面，大模型生成的标识符符合语法规则但缺乏语义连贯性[10]：人工代码变量命名与算法逻辑高度耦合、可自解释，大模型变量名多为机械组合，复杂函数实现中命名一致性显著下降。

注释风格差异揭示编程思维分野：人工注释采用“意图说明 - 实现细节”层次化结构，与代码逻辑互补；大模型注释呈“语句翻译”特征，仅复述表面功能，该差异直接影响初学者代码理解深度。

表 1 人工编写代码与大模型生成代码区别

人工编写代码	大模型生成代码

//在链表头部插入新节点
Struct Node* insertAtHead(struct Node* head, int data) {
// 1. 显式内存分配检查
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
if (newNode == NULL) {
fprintf(stderr, "Memory allocation failed\n");
// 错误处理标准化
exit(EXIT_FAILURE);
}
// 2. 分步初始化结构体成员
// 数据域赋值
newNode->data = data;
// 指针域连接
newNode->next = head;
// 3. 返回新头节点
// 单一出口原则
return newNode;
}
	// 链表头部插入函数
	struct Node* insert_front(struct Node* head, int val) {
	// 自动生成的紧凑写法
	struct Node* new_node = malloc(sizeof(*new_node));
	//未检查 malloc 返回值
	new_node->data = val;
	new_node->next = head;
	//直接返回结果
	return new_node;
	}

2.2 认知负荷与学习曲线影响

大模型辅助编程对初学者认知负荷的影响呈阶段性特征：语法学习初期能降低记忆负担，使学生聚焦算法逻辑而非语法细节，我校大一 C 语言教学数据显示，使用智能辅助工具的学生基础语法测试错误率降低约 8%，但复杂逻辑分析题表现比传统学习组差 11 个百分点。

传统编程学习遵循“语法-应用-实现”渐进路径，各阶段建立在前序知识基础上；大模型辅助学习呈“需求描述-代码获取-局部修改”跳跃特征，导致学生中期出现“能力断层”——独立完成中等复杂算法时，调试与错误处理能力弱于传统教学培养的学生。

人工编程时注意力集中于算法设计与错误预防，大模型辅助时认知焦点转移到需求表述与结果验证，虽提升短期任务效率，但不利于核心能力培养。

2.3 学适应性评估框架

本研究建立三维评估体系为课程内容筛选提供量

化依据：

代码可读性含结构清晰度、命名规范性、注释有效性三个二级指标，采用静态分析与专家评分结合的评估方式。实践显示大模型生成代码结构清晰度得分较高，但注释有效性低于人工教学范例，该差异在面向对象编程任务中更显著。

教学适配度考察代码示例的教育价值，评估要素包括概念完整性、错误示范合理性、思维启发价值。大模型生成代码多直接呈现最优解，缺乏 “从错误中学习” 的教学设计，需人工改造才能成为有效教学资源。

思维培养价值通过思维可视化程度、逻辑可追溯性、创新激发潜力综合评价，人工范例能清晰展现问题分解、模式抽象等计算思维要素，大模型代码功能完整但思维过程跳跃。基于此，基础教学阶段采用“混合展示” 策略，配对使用两类代码以兼顾技术先进性与教学有效性。

该框架应用表明，基础语法与算法核心逻辑需保留人工示范，工程实践与系统设计阶段可利用大模型多样化案例拓展学生视野。

3 课程重构模型设计

3.1 知识图谱构建方法

针对编程语言课程特点，设计分层递进知识图谱构建方法，将传统线性知识结构转化为多维关联网络：

基础层包含语法要素、数据类型、运算符等原子知识点，通过自然语言处理技术从教材和大纲中抽取概念实体，经教育专家校验形成标准化术语库。

关系层构建是创新重点，定义三类教学关联实现知识点连接：语法依赖关系（如指针依赖变量与内存地址理解）；逻辑推导关系（如循环与递归的等价转换）；认知关联关系（如数组与指针的异同）。该网络可动态反映学习者认知路径。

实践采用增量式构建策略：先建立含 34 个核心概念的基础图谱（如图 2 所示），再通过教学反馈扩展边缘节点。相关研究表明人工智能技术在 C 语言编程实验教学系统中已展现知识结构化支撑价值。

3.2 内容筛选与重组机制

以知识图谱为框架，结合大模型时代学习特征[11]，将编程语言知识点划分为三类模块：基础模块（变量定义、流程控制等语法核心）；思维强化模块（指针应用、递归算法等难点）；技术整合模块（文件操作、多线程等适合与大模型协作的内容）。

筛选引入双阈值评估策略：教学必要性通过专家评定（考量概念基础性、思维价值、迁移广度）；技术替代性通过实证研究统计 20 个典型任务中人工与大模型代码质量差异 —— 基础语法类任务人工优势度 12%，算法设计类仅 3%，该差异为模块划分提供量化支持。

重组采用 “核心稳固 - 边缘灵活” 原则：核心知识链维持传统线性序列以保障基础能力；边缘知识点允许动态组合（如函数教学中，参数传递机制固定教学路径，应用案例按学生兴趣选择），兼顾知识完整性与个性化需求。

3.3 混合式资源开发策略

针对大模型生成资源的教育适配性问题，提出三阶段混合开发策略：

资源准备阶段采用平行开发模式 —— 教学团队编写标准范例，同时生成大模型解决方案形成原始资源池。对比显示人工范例思维可视化得分比大模型方案高 18%，大模型方案在新颖性问题上覆盖更广、变体更多。

加工改良阶段实施专家标注机制，解决大模型资源教学适配问题，标注含思维过程注解、错误警示、扩展提示三个维度。实践表明改造后的大模型案例教学效果接近人工范例，学生理解度提升 8 个百分点；“思维对比” 注解通过并排展示两类方案差异，促进元认知能力发展。

应用实施阶段采用渐进式整合策略，按教学进度调整资源配比：语法基础阶段人工范例占 75%（强化概念理解）；算法训练阶段两类资源各占 50%（培养方案评估能力）；综合实践阶段大模型案例占 70%（侧重新颖案例）。该策略使班级在保持传统考核优势的同时，创新性项目表现优于对照组。

4 教学实践与效果评估

4.1 实验设计与实施过程

在我校（独立学院）计算机科学与技术专业 2024 级新生中开展对比教学实验，选取两个平行班级作为实验对象。实验组采用基于知识图谱的重构课程体系，对照组延续传统教学模式，两组学生入学成绩无显著差异。实验周期覆盖完整学期，共设置 16 个教学周，每周安排 2 学时理论授课和 2 学时实践训练。

实验组课程实施严格遵循三阶段教学模型。在语法基础阶段（第 1-4 周），采用限制性使用策略，仅允许学生在完成人工编程训练后，通过大模型生成的对比案例进行知识拓展。算法训练阶段（第 5-10 周）实施双轨制教学，每个知识点同时提供人工编写范例和大模型生成方案，要求学生分析两者在控制结构、

变量命名等方面的差异。综合实践阶段（第 11-16 周）则鼓励合理使用智能编程助手，重点培养在技术辅助下的系统设计能力。

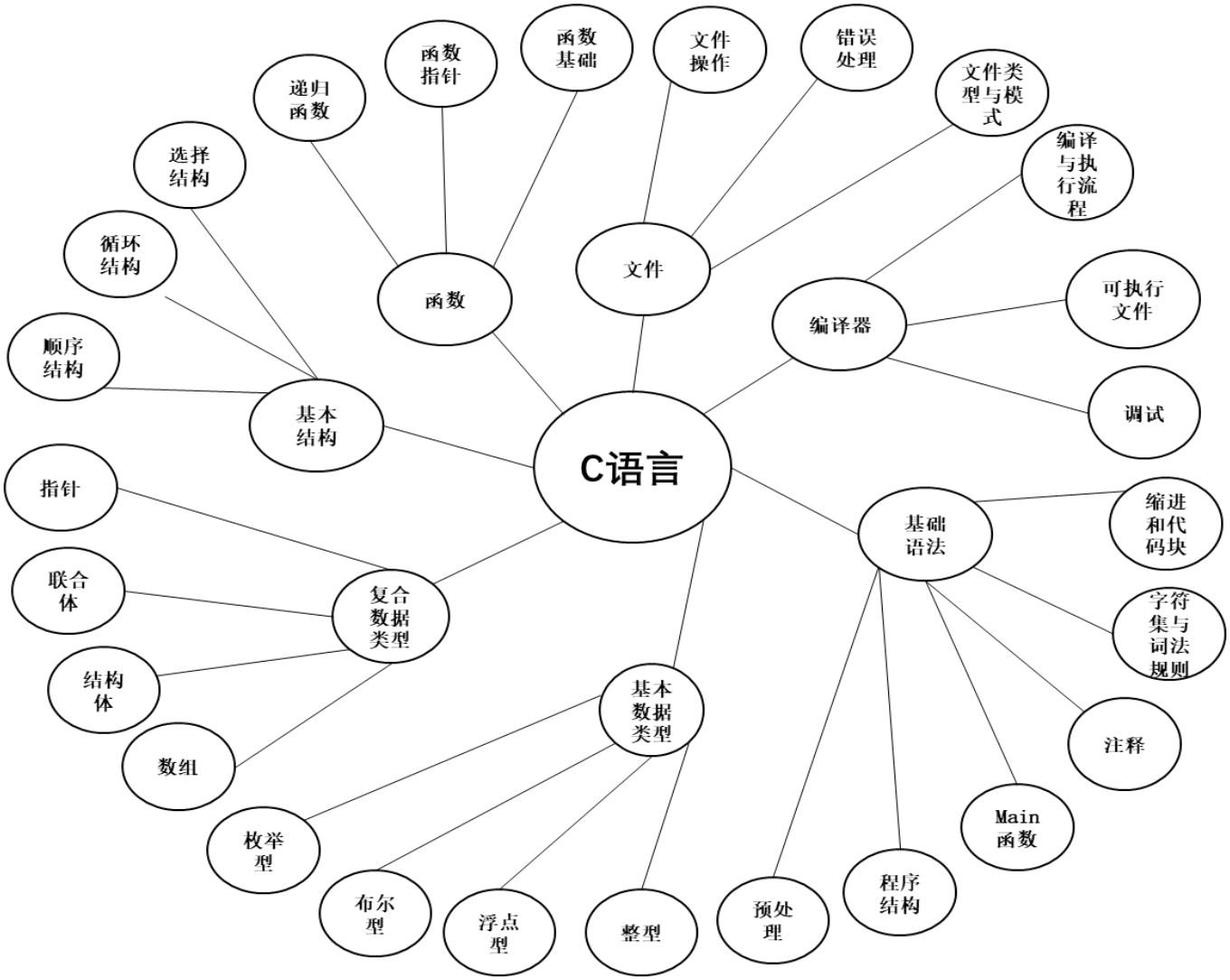

图 2 以 C 语言为例的部分知识图谱

4.2 多维效果评估分析

编程能力测试结果显示，实验组在基础语法掌握程度上与对照组无显著差异，但在算法设计题中表现出明显优势，平均得分高 11.2 个百分点。深入分析发现，实验组学生更擅长运用多重循环和递归等复杂结构，其代码在时间复杂度和空间效率上的优化程度显著优于对照组。

计算思维测评采用国际通用的 Bebras 测试工具，实验组在抽象能力、模式识别和算法思维三个维度均有提升。学习动机调查显示，实验组学生的自我效能感评分提高 5 个点，技术焦虑程度降低 15%。质性分析发现，实验组学生更倾向于将编程视为创造性活动，而对照组学生则更多表现出工具性学习倾向。

表 2 实验组与对照组语法掌握程度对比

测试项目	实验组（均值±标准差）	对照组（均值±标准差）	p值
基础语法题	88.5±6.2	85.3±7.1	0.12
多重循环嵌套题	76.4±8.5	65.2±9.3	0.003
指针概念理解题	82.1±7.8	70.6±10.2	0.001

行为数据分析揭示出更深层的教学效果。代码演变追踪显示，实验组学生在需求分析阶段耗时比对照组长 14%，但调试时间短 21%，表明其编程过程更具规划性和预见性。

表 3 计算思维测评（Bebras 测试）分数 0-100

维度	实验组	对照组

抽象能力	85	72
模式识别	78	65
算法思维	82	68
调试能力	80	70

大模型使用日志分析表明，实验组学生更注重通过提示词工程获取多样化解决方案，而非直接复制生成代码，这种差异在后期学习中逐渐扩大。

表 4　实验组学生大模型使用模式（周均次数）

行为类型	第1-4周	第5-10周	第11-16周
直接复制代码	3.2	1.8	0.5
生成对比方案	0.7	4.6	6.2
调试辅助	1.5	3.4	5.1

4.3　教学反思与改进建议

教学实践暴露出若干值得关注的问题。首先是大模型依赖的临界点现象，智能工具使用频率越高，学生的独立调试能力越下降。其次是思维惯性风险，部分学生倾向于机械套用大模型生成的代码模式，在面对非典型问题时表现出思维定势。此外还存在技术能力分化问题，约 35%的学生难以有效驾驭智能编程工具，反而增加认知负荷。

基于实践反馈提出三阶段改进策略。在入门阶段（1-4 周）应建立严格的使用规范，限制大模型仅用于答案验证而非代码生成。发展阶段（5-12 周）采用"人工优先"原则，要求先独立完成核心算法设计，再使用智能工具进行优化。熟练阶段（13-16 周）则引入"白盒使用"模式，强制要求学生解释大模型生成代码的关键逻辑，促进深度理解。

针对技术分化的应对措施包括分层任务设计和同伴辅导机制。将实践项目分为基础型（人工完成）和拓展型（允许智能辅助）两个难度层级，同时组织跨能力水平的学习小组。教师反馈机制也需要优化，建议采用实时分析仪表盘，动态监测学生的技术使用模式，及时干预过度依赖行为。

5　结束语

本研究提出的知识图谱驱动课程重构方法，通过三阶段技术整合策略，显著提升学生的算法设计能力（+11.2 分）和计算思维（抽象能力+13 分）。

核心创新在于：

① 建立教学必要性与技术替代性的双阈值筛选机制；

② 开发人工-生成双轨资源库（人工范例思维完整性+18%）；

③ 构建三维代码评估框架。

后续研究将聚焦：

① 基于实时行为数据的个性化适配系统开发；

② 代码演变过程的可视化评价工具设计；

③ 技术依赖临界点的动态监测（当前阈值 10 次/周）同时结合人工智能赋能教育的应用与创新趋势，探索跨学科融合的编程教育新范式。

参 考 文 献

[1] US Department of Education. Artificial Intelligence and the Future of Teaching and Learning: Insights and Recommendations [EB/OL]. (2023-05-01)[2024-03-20]. https://tech.ed.gov/ai-futureof-teaching-and-learning/

[2] 段继哲. 人工智能大模型下的自动生成代码——软件开发的新趋势[C]//《中国建筑金属结构》杂志社有限公司. 2024新质生产力视域下智慧建筑与经济发展论坛论文集（三）. 中远海运科技(北京)有限公司, 2024:18-19. DOI:10.26914/c.cnkihy.2024.043954.

[3] Šavelka, Jaromír et al. "Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses." *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (2023): n. pag.

[4] Arthur William Fodouop Kouam. "The Effectiveness of Intelligent Tutoring Systems in Supporting Students with Varying Levels of Programming Experience." Journal of Educational Computing Research (2024). DOI: https://doi.org/10.1007/s44217-024-00385-3.

[5] 蒲晓蓉，任亚洲, 杨阳，汤志伟. AI 大模型驱动高校人才培养改革[J]. 计算机技术与教育学报, 2024,12(06):101-105. DOI:10.12427/jcte2325-0208.20241217.

[6] 尹良泽, 徐建军, 李姗姗, 等. 人工智能时代下的计算机程序设计课程教学探索[J]. 计算机教育, 2025,(02):123-127. DOI:10.16512/j.cnki.jsjjy.2025.02.035.

[7] 王兵书, 李静怡, 雍珊珊, 李树一. 基于提示工程的程序设计探索与实践[J]. 计算机技术与教育学报, 2024, 12(3): 166-172.

[8] 王栋. 大语言模型在计算机编程实践课程教学中的策略与应用[J]. 信息与电脑, 2025, 37(11):239-241.

[9] 倪俊杰. 大模型编程的教学实践：课堂应用与创新升级[J]. 中国信息技术教育, 2025, (11):86-90.

[10] 段继哲.人工智能大模型下的自动生成代码——软件开发的新趋势[C]//《中国建筑金属结构》杂志社有限公司.2024新质生产力视域下智慧建筑与经济发展论坛论文集（三）.中远海运科技(北京)有限公司;2024:18-19.DOI:10.26914/c.cnkihy.2024.043954.

[11] 卢宇, 魏宁. 人工智能赋能教育：应用与创新[J]. 中国信息技术教育, 2025, (10):4-11.