生成式AI背景下程序设计类课程教学范式探索与实践*

张永忠* 赵玲玲 梁春霞

北京城市学院信息学部，北京 101309

摘 要 针对生成式 AI 背景下程序设计类课程教学离散、数据孤立、迁移评价缺失等问题，分析其可能因学生过度依赖 AI 而加剧思维惰性的风险，提出以“思维能力迁移”为核心的“教师协同—课程联动—任务驱动—制度保障”教学范式，介绍“百事通”智慧迁移平台及“三维五阶九指标”评价体系。实践表明，学生跨语言编程能力、解决综合性问题能力显著提升，省级以上竞赛获奖人数增长 32.8%。本文为生成式 AI 时代下程序设计类课程如何从“知识传授”走向“思维迁移”提供了系统化的理论框架与实践路径。

关键词 生成式 AI，程序设计课程，思维能力迁移，教学范式，智慧教学平台，迁移评价

Exploration and Practice of a Teaching Paradigm for Programming-Oriented Courses in the Context of Generative AI *

Yongzhong Zhang Linling Zhao Chunxia Liang

School of Information Science
Beijing City University,
znzx_bcu@163.com

Abstract—To address the problems of fragmented instruction, isolated learning data, and lack of transfer assessment in programming-oriented courses under the context of Generative AI, this paper analyzes the risk that students' over-reliance on AI may exacerbate thinking inertia. It proposes a teaching paradigm centered on "computational thinking transfer", consisting of instructor collaboration, curriculum articulation, task-driven learning, and institutional support. The "BaiShiTong" smart transfer platform and a "Three-Dimensions, Five-Levels, Nine-Indicators" evaluation system are introduced. Practice shows that students' cross-language programming ability and comprehensive problem-solving skills have been significantly improved, and the number of students winning provincial-level or higher competitions increased by 32.8%. This paper provides a systematic theoretical framework and practical pathway for shifting programming education from knowledge transmission to thinking transfer in the era of Generative AI.

Keywords—Generative AI, programming courses, computational thinking transfer, teaching paradigm,smart teaching platform, transfer assessment

1 引 言

• 生成式人工智能（Generative AI）的爆发式发展，正以前所未有的速度渗透到高等教育的各个领域。在程序设计类课程中，以Trace、ChatGPT、Cursor等为代表的AI编程助手，能够根据自然语言描述自动生成代码、解释错误信息、甚至完成跨语言的代码转换。这一技术变革在降低编程门槛的同时，也引发了教育者的深刻忧虑：当学生可以轻松获取现成代码时，他们是否还会认真思考循环的边界条件、递归的终止逻辑、指针与引用的本质区别？如果基本的编程思维训练被AI生成的代码所“短路”，学生将难以形成可迁移的核心能力[1]。

然而，生成式AI的出现并非程序设计教育困境的根源，而是放大了原有的问题。长期以来，程序设计类课程普遍存在“教学离散、数据断层、迁移评价缺失”三大痛点。不同编程语言课程（如C语言、Java、Python）分散在不同学期、由不同教师独立授课，课程之间缺乏思维层面的衔接[4]。学生在C语言中学到的循环、递归等核心思维，到了Java课程中往往需要“重新学起”，无法实现有效的跨语言迁移[2]。同时，学生的学习数据被割裂在各个课程平台中，后续课程的教师无法了解学生在前序课程中的掌握情况，个性化指导无从谈起。更重要的是，现有评价体系侧重于语法正确性和算法通过率，未将“思维能力迁移”作为独立的评价维度[5]。

*基金资助：本文得到北京城市学院教育科学研究课题资助，课题编号：JYC20251207

*通讯作者：张永忠 znzx_bcu@163.com 。

上述问题在生成式AI背景下变得更加紧迫。一方面，AI可以充当“即时翻译官”，快速将一段代码从一种语言转换为另一种语言，这似乎为迁移学习提供了便利；另一方面，如果教学设计不当，学生可能直接使用AI绕过对比、映射等认知过程，形成“思维依赖”而非“思维迁移”[3][8]。因此，如何系统设计教学范式、平台工具和评价机制，使学生既能利用AI提高学习效率，又能保持并发展可迁移的编程思维，成为亟待解决的核心问题。

针对上述挑战，自2018年起，我们逐步提出，并建成了“思维能力迁移”教学理念，构建了完整的教学范式，研发了智慧迁移平台，建立了可量化的评价体系。在生成式AI快速发展的新阶段，我们坚持“以思维能力迁移”为不变核心，将AI工具定位为辅助迁移的支架，而非替代思维的工具。本文将详细阐述我们的理论框架、技术实现与实践效果，以期为生成式AI时代的程序设计教育改革提供参考。

2 程序设计类课程思维能力迁移的困境分析

2.1 教学离散：课程孤立导致思维联结断裂

程序设计思维的核心要素包括数据抽象、控制流、递归、模块化、并发等。这些要素在不同编程语言中虽有语法差异，但其逻辑本质是相通的[2]。然而，当前高校的程序类课程设置却呈现出显著的离散化特征。如表1所示，C语言、Java、Web前端、Python等课程被分散在大一至大二的不同学期，通常由不同教师独立授课。每门课程聚焦本语言的语法细节，而不同课程之间缺乏对共性思维要素的萃取与联结[4]。

表 1 典型高校程序类课程离散开课情况

开课学期	课程名称	编程语言	授课教师
大一上学期	程序设计基础	C 语言	教师 A
大一下学期	面向对象程序设计	Java	教师 B
大二上学期	Web 前端开发	HTML/JavaScript	教师 C
大三上学期	Python 科学计算	Python	教师 D

这种离散化设置导致学生形成了“语言中心”而非“思维中心”的学习习惯。学生往往将不同语言视为相互独立的知识体系，而未能意识到循环、递归等思维要素在不同语言中的映射关系。当从C语言进入Java时，学生需要重新学习for-each循环的写法，却不理解其在逻辑上与C语言循环的同构性。这种“重复低效”的教学模式，不仅浪费了学习时间，更重要的是阻碍了学生形成可迁移的思维结构[2][4]。

2.2 数据断层：学习记录割裂导致迁移路径中断

迁移不是一次性的动作，而是一个需要持续跟踪和支持的认知过程。理想的迁移教学要求后续课程的教师了解学生在前序课程中的具体表现：该生在递归问题上是否存在困难？在调试效率上处于什么水平？哪些思维要素已经掌握，哪些仍需强化？然而，当前的课程平台各自为政，学生的课堂测试、课后作业、代码提交记录等过程性数据被隔离在不同系统中，形成“数据孤岛”。

表2从三个维度对比了理想迁移教学与现实状况的差距。数据断层的直接后果是：后续课程的教师只能从零开始，无法进行针对性的迁移衔接教学；学生也失去了获得连续性反馈的机会。更为严重的是，当学生进入新课程时，前序学习中已经形成的迁移基础（或迁移障碍）被完全忽略，导致迁移通道在起点处就被中断[6]。

表 2 理想迁移教学与实现状况对比

维度	理想状态	现实状况
前序知识传递	后续教师了解学生的能力画像	数据隔离，教师无从知晓
个性化指导	针对薄弱环节推送迁移任务	统一教学，无法因材施教
迁移路径追踪	记录学生跨课程的学习轨迹	轨迹断裂，无法追踪

2.3 评价缺失：迁移能力缺乏量化衡量标准

评价是指挥棒。传统的程序设计课程评价体系侧重于语法规则记忆和算法正确性验证，考核内容大多

限定于当前课程所授语言。对于“学生能否将一种语言的思维迁移到另一种语言”这一核心能力，现有评价几乎没有涉及[5][7]。

具体而言，有三个层面的缺失：

其一，迁移过程中的要素抽取、结构映射、问题转化等关键认知行为无法被捕捉；

其二，教师无法追踪学生的迁移路径，难以诊断迁移障碍出现在哪个环节；

其三，学生缺乏对自身迁移能力的量化认知，无法进行有针对性的自我提升。

上述三大困境相互关联、互为因果。教学离散导致思维联结断裂，数据断层使迁移指导无法个性化，评价缺失则让所有改进努力缺乏方向。这正是本成果试图系统解决的核心问题。

3 思维能力迁移教学范式的构建

针对上述困境，本文提出以“思维能力迁移”为核心的教学理念，并构建了“教师协同—课程联动—任务驱动—制度保障”的四维教学范式，如图1所示。

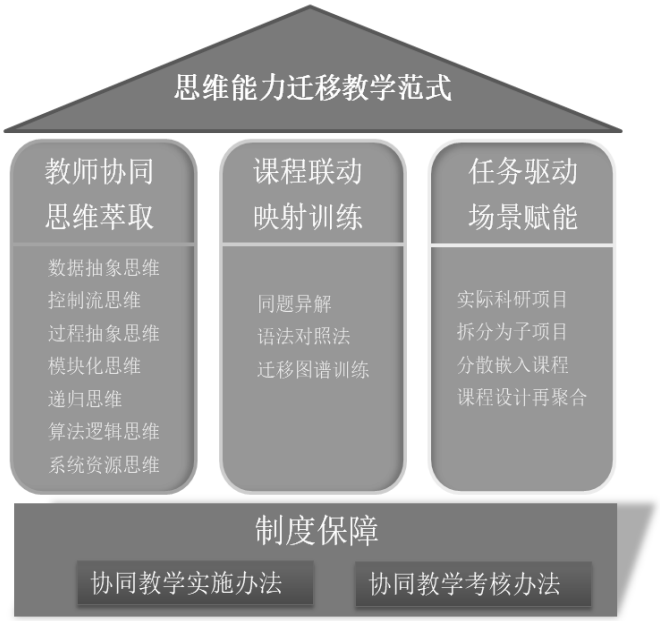


图 1 思维能力迁移教学范式框架图

3.1 教师协同：从“单兵作战”到“团队萃取”

迁移教学的第一步是建立跨课程的教师协同机制。本成果组建了涵盖C语言、Java、Python、Web前端等课程的教师团队，通过定期教研活动，共同分析不同语言在数据表示、控制结构、函数定义、异常处理、模块组织等方面的异同。团队编制了《迁移教学手册》，该手册包括三部分内容：

（1）核心思维要素清单，共7大类23个要素；

（2）多语言语法对比表，直观展示同一逻辑在不同语言中的实现差异；

（3）知识迁移图谱，标注前序课程与后续课程之间的知识点映射关系。

教师协同使迁移教学有了共同的认知基础，避免了因教师个人差异导致的随意性。

3.2 课程联动：设计“同题异解”项目链

有了思维要素萃取成果后，教学团队设计了贯穿多门课程的“同题异解”项目链。具体做法是：选择若干具有代表性的算法问题（如排序、查找、数据统计、图形绘制等），要求学生在不同课程中分别用当前所学语言实现。每次实现时，教师引导学生回顾之前语言中的解法，对比语法差异，找到逻辑上的对应关系。

以“递归”这一思维要素为例。在C语言课程中，学生用递归实现斐波那契数列；到Java课程时，教师不仅要求学生再次实现，还要对比Java的方法调用与C语言函数的异同，并讨论尾递归优化在不同语言中的支持情况。通过这种“对比—映射—应用”的反复训练，学生逐渐建立起“算法逻辑是内核，语法只是外壳”的认知，形成可迁移的思维模型。

3.3 任务驱动：企业项目“拆解—嵌入—聚合”

为了将迁移能力训练延伸到真实工程场景，本成果采用了“化整为零—分布嵌入—再聚合”的任务驱动策略。选取一个中等规模的企业级项目（如“物联网数据采集平台”），将其拆解为多个独立但相互关联的子任务。这些子任务按照难度和技术栈要求，分别嵌入到不同学期的程序设计课程中作为课程大作业。当学生完成所有子任务后，在最后一门课程（或专门的综合实践课）中，将这些子任务重新聚合为完整项目。此时，学生需要综合运用多门课程中学到的知识和技能，完成跨语言、跨模块的集成，如图2所示。这种真实场景下的任务驱动，极大地锻炼了学生的应用迁移能力。

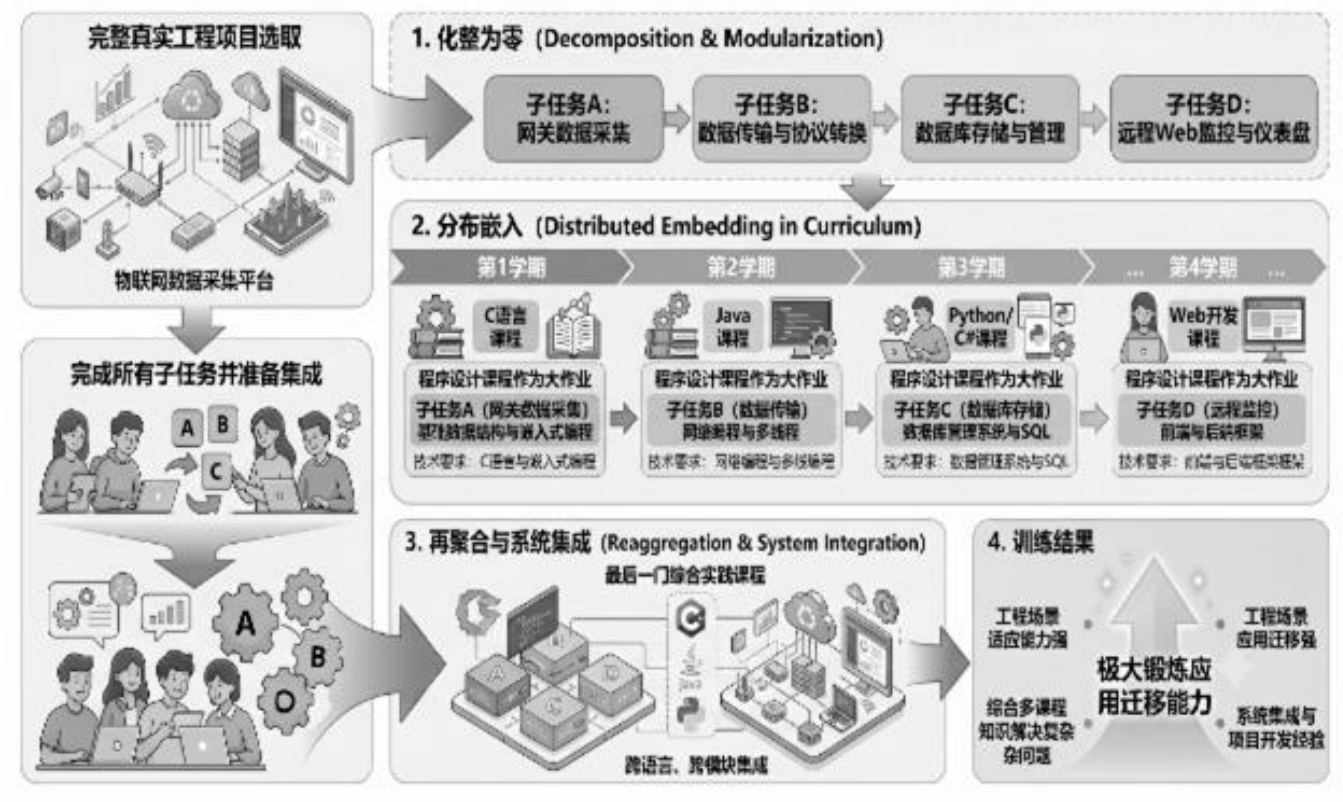
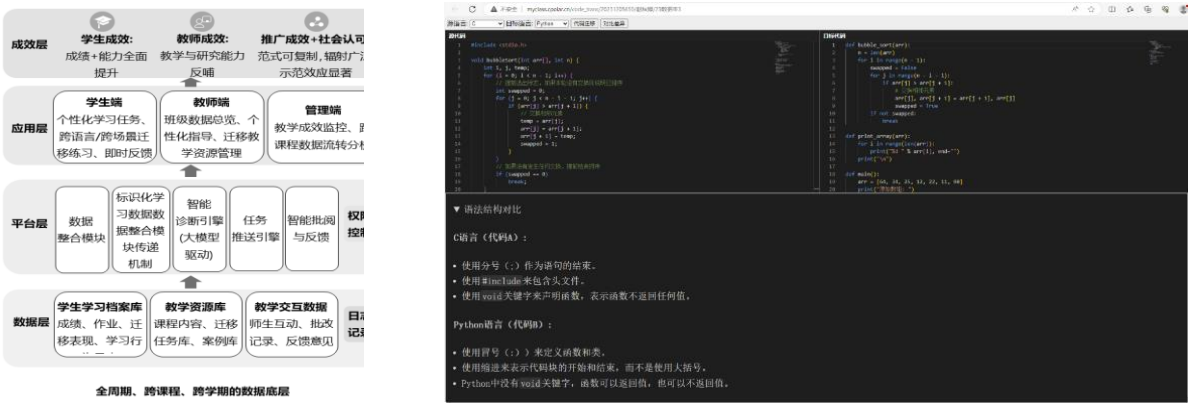


图 2 基于真实工程场景的“化整为零-分布嵌入-再聚合”任务驱动



a）平台架构图　　b） 代码迁移

图 3 百事通智慧迁移平台

3.4 制度保障：固化协同教学机制

为确保迁移教学的可持续性，本成果制定了《程序设计类课程协同教学实施办法》和《迁移教学考核办法》。前者规定了跨课程教研活动的频率（每学期至少4次）、迁移手册的更新机制、学生数据共享的规范等。后者将教师参与迁移教学的情况（如编写对比案例、指导学生迁移作业等）纳入教学工作量考核。制度保障使迁移教学从“自发行为”上升为“规范动作”，避免了因人而异的随意性。

4 智慧迁移平台的设计与实现

教学范式需要平台工具的支持才能落地生根。本成果自研了“百事通”智慧迁移教学平台，其核心目标是：打通跨课程、跨学期的学习数据链路，并为迁移过程提供智能化支持。平台整体架构如图3所示。

4.1 数据采集与存储层

平台底层建立了统一的“学生学习档案库”，该库以学生为中心，汇集其在整个大学阶段所有程序设计课程中的学习行为数据。数据来源包括：课堂互动

（随堂测验、提问）、课后作业、在线实验（代码自动评测）、阶段考试、项目实践以及教师的主观评价。所有数据通过“数据标签化引擎”进行处理，被转化为带有统一语义的标签，如“循环掌握度=0.85”“递归掌握度=0.32”“调试效率=中”。

4.2 数据联通与传递层：标识化学习数据传递机制

这是平台的核心创新。传统教学平台的数据是“按课程分库”存储的，课程之间没有关联。而本平台设计了“标识化学习数据传递机制”：当学生完成一门课程后，系统自动将其学习标签进行抽象和编码，形成一个“迁移起点向量”。当学生开始下一门课程时，该向量被传递到新课的教学空间中，后续课程的教师可以看到前序课程的能力画像，系统也会基于该向量生成个性化建议。

该机制的具体运作流程如下（以某生为例）：

在“程序设计基础（C语言）”课程中，系统记录：循环掌握度=0.8，递归掌握度=0.3，迁移速度=0.5。

进入“面向对象程序设计（Java）”课程时，平台检测到递归能力薄弱，自动推送“Java递归专项训练”任务；同时因迁移速度较低，在课程初期分配额外的迁移图谱学习任务。

学生完成任务后，系统更新档案：递归掌握度=0.7，迁移速度=0.6。

这一机制从技术上解决了数据断层问题，实现了学生轨迹的“伴随式流转”。

4.3 智慧迁移支持层

在数据贯通的基础上，平台集成了多项智能化功能来直接支持迁移教学过程。

（1）智能推送引擎：根据学生当前的能力标签和历史迁移表现，推荐个性化的练习题和学习资源。

（2）语法对比与映射工具：提供C、Java、Python三种语言的代码并排对比功能，学生可以高亮显示不同语言中实现同一逻辑的语句对应关系。

（3）智能批阅与诊断系统：对学生的迁移任务作业进行自动批改，不仅判断正确性，还能定位思维迁移中的具体问题（如“循环边界理解正确，但数组索引习惯未迁移”）。

4.4 教学管理与应用层

最上层为教师和管理者提供了可视化工具。教师可以查看所授课程全体学生的迁移能力分布图、迁移路径图、能力雷达图等。通过“教师协同模块”，不同课程的教师可以就某个学生的特殊情况发起讨论或交接。学习效果追踪面板则实时更新每位学生的评价结果，为教学干预提供依据。

5 思维能力迁移的评价机制

5.1 传统评价的局限与迁移评价的维度

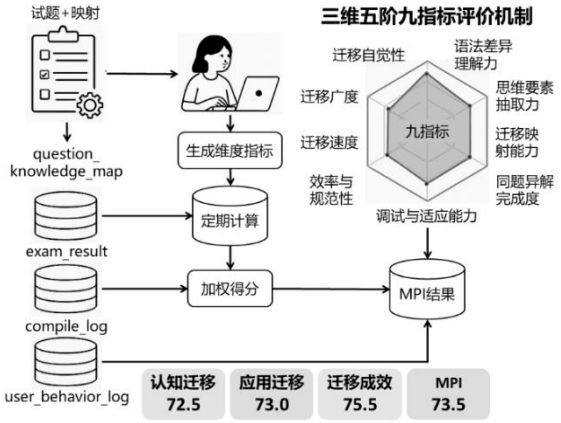
传统的程序设计课程评价存在三个根本局限：

一是评价内容局限于单一语言，不涉及跨语言比较；

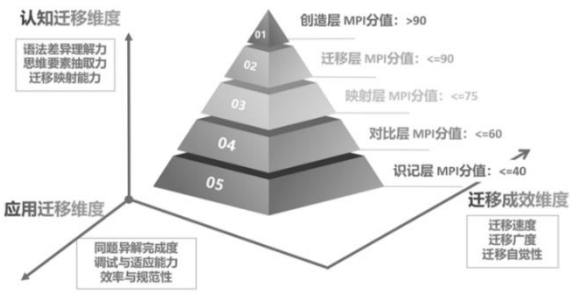
二是评价指标局限于正确性，不涉及迁移过程的质量；

三是评价方式局限于结果导向，不涉及迁移能力的成长轨迹。

为此，本成果设计了“三维—五阶—九指标”的迁移评价机制，如图4所示。



a）评价机制



b）评价等级

图 4 “三维五阶九指标”评价机制图

5.2 三维度：认知迁移、应用迁移、迁移成效

评价体系从三个维度综合衡量学生的迁移能力。认知迁移维度评估学生对不同语言思维要素异同的理解程度，包括能否正确识别、对比、解释不同语言中的对应结构。应用迁移维度评估学生能否在给定任务中实际使用一种未系统学过的语言（或一种语言的新特性）解决问题，包括代码的正确性、风格、调试过程等。迁移成效维度评估迁移的速度、广度和自觉性，例如学生从掌握C语言到能够使用Java完成相同任务所需的训练量。

5.3 五层次：从识记到创造

借鉴布鲁姆教育目标分类学，将迁移能力的发展划分为五个递进的层次。识记：能记住单种语言的语法规则。对比：能在教师指导下找出两种语言在实现同一功能时的语法差异。映射：能独立地将一种语言中的代码逻辑“翻译”成另一种语言。迁移：面对新问题，能主动选择合适语言，并快速完成从已有知识到新语言的转换。创造：能综合多种语言的优势，设计跨语言的系统解决方案。每个层次都有对应的行为锚定描述，便于教师评判。

5.4 九指标与迁移指数（MPI）

为便于数据采集和自动计算，上述维度和层次被具体化为九个指标：（1）语法差异理解力；（2）思维要素抽取力；（3）迁移映射能力；（4）同题异解完成度；（5）调试与适应能力；（6）效率与规范性；（7）迁移速度；（8）迁移广度；（9）迁移自觉性。平台通过分析学生的代码提交、实验报告等数据，自动计算这九个指标的得分，并加权合成“迁移指数（MPI）”。

该评价机制的价值不仅在于“打分”，更在于驱动教学改进。学生可以随时查看自己的迁移能力雷达图，了解薄弱环节，并获得个性化的学习建议。教师可以查看班级的整体迁移分布，针对共性问题组织专题讲解。由此，“教—学—测—评”形成了一个数据驱动的闭环系统。

6 教学实践与应用效果

本成果在计算机科学与技术、软件工程、物联网工程、机电工程等专业中进行了为期三年的系统实践。以下从学生成绩、竞赛、科研、教师发展、推广辐射五个维度报告应用效果。

6.1 学生成绩与能力的提升

对成果实施前（2019级）与实施后（2022级）学生的学习成绩进行分析，结果显示：实施后学生在多门程序设计课程上的平均分均高于实施前，最高提高了6分。图5展示了成绩比较情况。

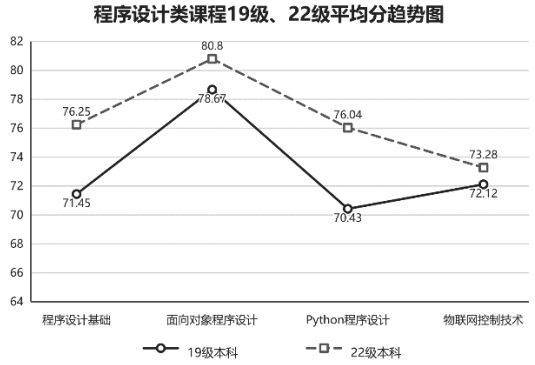


图 5 程序设计类课程 19 级、22 级平均分比较

在学科竞赛方面，省级及以上竞赛获奖总人数从116人（其中国家级38人、省部级78人）增长至154人（国家级41人、省部级113人），增长32.8%。含金量较高的省级一等奖从30项增至54项，国家级一等奖从5项增至6项。图6展示了竞赛获奖数量的变化。

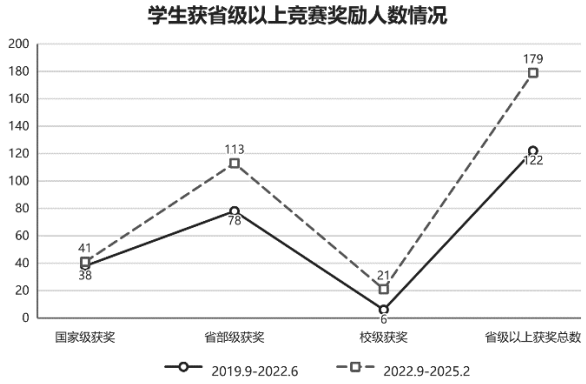


图 6 学生获省级以上竞赛奖励数量

在科研产出方面，学生参与程序类科研项目的数量从3项增至17项，增长约467%；软件著作权申请量从1项增至7项，增幅达600%。在工程实践能力方面，学生团队作为核心开发人员，参与了增强现实、微电子产线控制等10余个航空航天项目，能够跨语言（C++/Python/Java）承担主要开发工作。

6.2 教师能力提升与成果产出

成果实施带动了教师团队的成长。团队成员获评北京市青年教学名师1人；荣获北京市教学创新大赛、优质本科课程等省部级奖项11项；出版教材7本，其中《程序设计思维迁移实训教程》被5所兄弟院校选用；

相关教学研究论文获中国计算机实践教育学术会议优秀论文奖。

7 结语

7.1 主要结论

本文针对程序设计类课程中长期存在的“教学离散、数据断层、迁移评价缺失”三大问题，系统提出了以“思维能力迁移”为核心的教学解决方案。主要结论如下：

第一，“思维能力迁移”可以作为程序设计类课程的核心教学目标。不同于传统教学侧重于单一语言的语法与算法，本成果证明学生能够在教师引导下建立跨语言的思维映射，形成可迁移的编程思维结构。

第二，“标识化学习数据传递机制”是打通跨课程数据孤岛的有效技术路径。通过将学生学习表现抽象为统一标签并伴随其学习进程传递，可以实现个性化迁移指导和连续性的学习追踪。

第三，“三维五阶九指标”评价体系能够量化学生的迁移能力。迁移指数（MPI）的引入，使迁移从“模糊的感受”变为“可测量的指标”，为教学改进提供了数据依据。

7.2 生成式AI背景下的启示

在生成式AI快速发展的背景下，本成果的核心理念——培养学生可迁移的编程思维——显得尤为重要。AI工具可以成为迁移学习的有效支架（如提供多语言对比示例、生成个性化迁移任务），但前提是学生必须具备自主的思维迁移能力，才能对AI输出进行批判性评估和创造性修改。如果学生没有经过系统的迁移训练，AI反而会成为思维的替代品。因此，生成式AI时代的程序设计教育，比以往任何时候都更需要“思维能力迁移”这一教学理念的引导。

7.3 研究局限与未来方向

本研究存在若干局限。首先，迁移指数（MPI）的权重设置目前基于专家经验，未来需要更多数据驱动的优化。其次，本成果主要在理工科专业中应用，在文科类专业的编程教育中的适用性有待验证。未来，我们将进一步探索生成式AI与迁移教学的深度融合，研究如何利用大模型自动生成个性化迁移路径，并开发更加智能的迁移诊断工具。

参 考 文 献

[1] Becker B A, Denny P, Finnie-Ansley J, et al. Programming is hard - or at least it used to be: Educational opportunities and challenges of AI code generation[C]//Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE 2023). ACM, 2023: 500-506

[2] Kao Y, Matlen B J, Weintrop D. From one language to the next: Applications of analogical transfer for programming education[J]. ACM Transactions on Computing Education, 2022, 22(4): 1-21

[3] Sun D, Boudouaia A, Zhu C, Li Y. Would ChatGPT-facilitated programming mode impact college students ‘ programming behaviors, performances, and perceptions? An empirical study[J]. International Journal of Educational Technology in Higher Education, 2024, 21: 14

[4] 嵩天, 黄天羽, 黄为民. Python 语言程序设计教学中的计算思维迁移研究[J]. 计算机教育, 2021(1): 23-27

[5] 殷红, 陈萍, 张莉. 程序设计类课程群建设中的能力迁移路径探索[J]. 软件导刊, 2022, 21(5): 112-116

[6] 王立, 李海. 基于知识图谱的程序设计课程迁移学习平台设计与应用[J]. 现代教育技术, 2023, 33(6): 68-75

[7] 陈娟, 邓磊. “计算+X”复合型人才培养中编程思维迁移的实践研究[J]. 高等工程教育研究, 2024(3): 56-61

[8] 刘强, 赵一鸣. 生成式 AI 对计算机专业学生编程迁移能力的影响及对策[J]. 计算机教育, 2025(2): 34-39