Vibe Coding 无代码赋能大学计算思维培育与融合教学实践*

夏小俊[1,3]** 姚嘉轩[2,3]

1.东南大学生物科学与医学工程学院，江苏南京 211189

2.东南大学软件学院，江苏南京 211189

3.东南大学吴健雄学院计算思维与程序实践课程研究中心，江苏南京 211189

摘 要 针对传统C++课程语法门槛高、难以系统培育计算思维，以及AI时代Vibe Coding范式引发学生核心编程技能衰退的风险，本文设计了融合传统教学与Vibe Coding的程序设计课程改革方案。在坚持C++核心语法不删减原则下，以EasyX图形库为中介构建了“目标-流程-案例-评价”四位一体教学模型，实施结构化双轨时间线课内教学，并建立了四维协同评价体系。基于82名本科生的准实验研究结果显示，相较于对照班，实验班的AI协同认知均值从3.46提升至4.35，机考平均分从75.2分提高到78.9分，优秀率由25.0%提升至31.0%。该模式在不削弱传统编程能力的前提下有效降低了入门难度并培育了计算思维，为同类课程改革提供了可复制范式。

关键字 Vibe Coding；无代码编程；计算思维；C++程序设计；AI辅助编程

No-Code Empowerment of Computational Thinking Cultivation and Integrated Teaching Practice in Universities through Vibe Coding *

Xia Xiaojun[1,3]** Yao Jiaxuan[2,3]

1. School of Biological Science and Medical Engineering, Southeast University, Nanjing 211189, China
2. School of Software Engineering, Southeast University, Nanjing 211189, China
3. Research Center for Computational Thinking and Programming Practice Curriculum Chien-Shiung Wu College, Southeast University, Nanjing 211189, China

Abstract—Aiming at the high syntactic threshold of traditional C++ courses that hinders systematic cultivation of computational thinking, as well as the risk of students' core programming skill decline triggered by the Vibe Coding paradigm in the AI era, this paper designs a curriculum reform scheme for programming design that integrates traditional teaching with Vibe Coding. Adhering to the principle of retaining the core syntax of C++, a four-in-one teaching model encompassing "objectives, processes, cases, and evaluation" is constructed using the EasyX graphics library as a visual intermediary. Structured dual-track timeline in-class teaching is implemented, and a four-dimensional collaborative evaluation system is established. The quasi-experimental research results based on 82 undergraduate students show that, compared with the control class, the mean value of AI-assisted cognitive recognition in the experimental class increased from 3.46 to 4.35, the average score of the computer-based exam rose from 75.2 to 78.9, and the excellence rate escalated from 25.0% to 31.0%. Without weakening traditional programming abilities, this model effectively reduces the difficulty of entry and cultivates computational thinking, providing a replicable paradigm for the reform of similar courses.

Keywords—Vibe Coding; no-code programming; computational thinking; C++ programming; AI-assisted programming

1 引 言

长期以来，程序设计语言类课程（以C++语言为例）一直是我国高校计算机类、电子信息类等理工科专业的必修课程，始终扮演着培育学生计算思维、逻辑思维以及底层工程素养的核心角色。在经典的教学模式中，课程以语法讲解、控制台程序编

*基金资助：本文得到2025年东南大学“AI-MUST”课程建设项目，2025年东南大学课程思政教改项目。

**通讯作者：夏小俊 xxj.rcls@seu.edu.cn。

写、逐行代码调试为主要形式，强调对语言规则的记忆与算法熟练度训练。然而这种模式暴露出一系列难以回避的局限：C++语法结构复杂、细节约束严格，初学者容易陷入语法错误排查而难以把握程序整体逻辑；程序输出以文本为主，运行效果不直观，缺乏即时反馈，难以激发学习动机；教学重心偏向代码书写，对问题拆解、抽象建模、流程设计等计算思维核心环节训练不足；教学环境与当前真实工程环境存在差距，学生缺乏 AI 协同、组件化开发、可视化设计等现代工程意识。

另一方面，随着大模型、智能体、Skills 等新兴人工智能技术的兴起，以 Vibe Coding、AI 辅助编程、低代码/无代码平台为代表的新型开发方式正在深刻改变现实中的软件开发流程。开发者很多时候已不必逐行手写代码，而是以自然语言清晰描述需求，由 AI 自动完成代码生成、补全、纠错与优化。这种模式更贴近人类思考问题的自然路径，能够有效降低实现门槛，尤其适合用于帮助初学者建立“需求—逻辑—实现”的完整思维链条。

但新技术的狂飙突进不可避免地引发了教育圈的焦虑与学术争论。如果高校程序设计教育全面拥抱 Vibe Coding，允许学生过度依赖 AI 生成代码，将不可避免地导致学生底层逻辑理解的缺失、核心编程技能的衰退以及对计算机系统底层运行机制的无知；但若固步自封，坚决将 AI 工具拒之门外，则会使学生严重脱离现代软件工业的实际发展趋势与未来岗位需求。因此将传统教学方式与 Vibe Coding 范式进行适度、有序、可控的融合，将当前高校程序设计课程改革的重要方向[1]。

2 从传统编码到 Vibe Coding，效率跃升与隐藏的风险

传统编程范式下，开发者必须精通特定编程语言（如 C++）的语法、数据结构与内存分配机制。这种模式要求开发者具备极高的精确控制力与逻辑严密性，每一个函数调用、每一次指针寻址都必须经过缜密的设计。传统编程在确保系统安全性、执行效率和长期可维护性方面具有不可替代的优势，是构建复杂底层架构的基石。然而，这种“代码中心化”（Code-Centric）的过程异常耗时，学习曲线极为陡峭，且容易将学习者的认知带宽大量消耗在语法细节与繁琐的调试追踪上，而非宏观的业务逻辑与创新设计上。

2025 年初，人工智能学者 Andrej Karpathy 正式提出了“Vibe Coding”（氛围编程/意图式编程）这一全新概念。该概念描绘了一种革命性的软件开发工作流：开发者将主要精力从逐行编写枯燥的源代码，彻底转移到通过自然语言提示词（Prompt）与 AI 编程助手进行对话式交互，由 AI 模型自动完成应用程序的生成、细化、重构与调试。这一范式的出现，在极大降低编程门槛的同时，赋予了非技术背景人员强大的原型开发能力，引发了全行业的广泛应用与热烈探讨[2]。

但是在将 AI 生成技术引入课堂之前，教育管理者与一线教师必须对该技术可能带来的负面效应保持高度的学术清醒。大量 2025 至 2026 年间的实证研究揭示，无节制地使用 Vibe Coding 不仅无法提升学生的真实编程能力，反而会引发严重的“认知卸载”（Cognitive Offloading）与技术债务累积。如 2025 年 12 月，知名代码审查平台 Code Rabbit 对 470 个开源 GitHub 拉取请求（PR）进行了一项深度分析。结果发现，由 AI 协作生成的代码块中包含的“重大”问题数量是纯人类编写代码的 1.7 倍。具体而言，涉及不正确依赖关系、控制流缺陷及配置错误的逻辑错误高出人类代码 75%；更为严重的是，AI 生成代码引入的安全漏洞率是人类的 2.74 倍。同时，Veracode 发布的《2025 年生成式 AI 代码安全报告》证实，约 45%的 AI 生成代码样本无法通过基础的安全测试，并包含大量关键漏洞。这表明，如果不具备扎实的传统语言功底以进行严密审查，直接使用 AI 代码将带来灾难性的工程后果。

3 面向计算思维培育的 C++融合教学设计

3.1 计算思维培养的四个层次

首先我们仍然要明确，无论采用何种编程范式，程序设计类课程最本质的目标仍然是培养学生的高阶思维能力，特别是计算思维和工程思维等计算思维是运用计算机科学基本概念解决问题、设计系统、理解行为的普适性思维活动，核心包含问题分解、抽象建模、模式识别、算法设计、自动化实现与调试优化五大维度，其培养核心是让学生学会用计算机科学思维解决问题，而非单纯训练编程技能[3]。

在人工智能时代，计算思维的内涵可以进一步升华[4]，在我们的研究当中将其分为四个层次：

（1）基础层（程序理解与逻辑可视化表达）。此层面旨在帮助入门阶段的学生跨越心理门槛。要求学生不仅能够读懂 C++的变量声明、数据类型以及顺序、分支、循环三大基本控制结构，更要求其具备使用标准流程图（Flowchart）将自然语言需求转化为结构化逻辑的可视化表达能力。这一层面的核心在于消除对代码的恐惧，建立起“逻辑可表达、程序可理解”的底层信念。

（2）核心层（经典计算思维能力深化）：这是整个培育体系的中枢，全面覆盖问题求解的完整生命周期。学生必须具备问题分解能力、抽象建模能力、算法设计能力、数据处理能力等核心关键能力。

（3）工程层（代码规范与严谨工程意识）：这一层级旨在将学生从“能写代码的代码工”升华为“能写好代码的工程师”。教学重点转向变量命名规范、注释撰写逻辑、代码缩进排版，以及最为关键的自主错误排查与程序性能优化能力。建立起对技术债务的敬畏之心，是走向专业开发不可或缺的阶梯。

（4）创新层（现代开发范式与人机协同思维）：作为 AI 时代的特色培养目标，要求学生具备 Vibe-Coding 维度的可视化与参数化思维，能够熟练运用提示词工程与 AI 协作，独立驾驭从需求痛点分析、逻辑图谱拆解、人机协同编码到图形界面呈现与用户体验优化的全栈式现代开发流程。

表1　融合传统教学与Vibe Coding的教学设计表示例

教学阶段	传统知识体系	Vibe Coding 融合	计算思维培育目标
计算认知启蒙	语言基础结构、数据类型、变量作用域、算术与逻辑运算符、控制台 I/O 交互	引入流程图绘制工具实现逻辑链路可视化；演示 AI 对初级语法错误的诊断与原理解释；模拟基础的表单交互与数据流转	建立程序架构整体认知；消除语法恐惧感；初步体验人机协同调试模式的效能。
逻辑控制与算法抽象建模	单分支(if)、多分支(switch)、while 与 for 循环、复杂的循环嵌套逻辑	利用图形化思维拆解复杂判断条件；借助 AI 进行极值边界测试与死循环沙盘推演；将抽象循环映射为 EasyX 中的几何图形重复绘制过程	深度强化问题分解与算法设计能力；在动态逻辑执行中培养严密的条件判断与流程控制心智。
系统解耦与组件化工程思维	函数声明与定义、作用域隔离、值传递机制的内存表现	将函数的概念与 No-Code 平台中的“可拖拽独立组件”进行哲学类比；通过修改函数入口参数实时观察图形渲染变化	培育抽象建模能力；建立“高内聚低耦合”与“一次定义、多次调用”的现代工程组件化设计理念。
底层数据组织与内存寻址逻辑	一维与多维数组、地址传递、指针的声明与解引用、指针函数	借助可视化工具在屏幕上模拟内存条的物理排布与指针箭头的指向跳转；利用大语言模型将深奥的指针语法转化为生活化场景比喻	攻克 C++最为艰深的底层逻辑堡垒；提升对内存空间、数据存储连续性及寻址效率的底层认知维度。
全栈项目综合实战与效能跃迁	前述所有核心语法的综合融会贯通、EasyX 图形库各类高级 API 的统筹调用	独立承担综合项目：完成需求文档撰写、模块化协同编码、全息可视化呈现及基于用户体验的参数化效果持续迭代	综合检验四层培育目标；彻底落实智能体协同设计理念与参数化架构优化能力。

3．2　融合传统方式与 Vibe Coding 的教学设计

在本研究当中，Vibe Coding 并非指完全不教或者不写代码，而是一种融合式教学理念：以无代码思维帮助学生理解程序逻辑，以 Vibe Coding 工具辅助学习、纠错与优化，并辅以 EasyX 图形库搭建抽象逻辑与直观效果之间的桥梁，最终在保留 C++核心代码训练的基础上，真实提升学生的计算思维与工程素养[5]。

为保证教学目标的达成，在课程的融合当中确定四项基础原则。第一，代码能力不弱化，保留 C++所有核心知识点，要求学生必须手写关键代码，避免完全通过工具替代思考。第二，思维训练更突出，教学重心从“写对代码”适度转向“学会思考、学会拆解、学会设计”，强化计算思维全过程。第三，

工具辅助不替代，AI 与 No-Code 工具仅用于理解、调试、优化与拓展，不替代学习过程与独立思考。第四，评价可量化可操作，以量化指标保证评价公平，以质性反馈促进学生持续改进，形成教学闭环。

为了确保传统语言教学的稳固与新范式思维的平滑注入，课程在时间维度上进行了精心设计。在每节 45 分钟的标准课时内，划定结构化双轨时间线：前半部分作为“传统知识高压舱”，由教师主导进行 C++核心语法的板书推演、内存机制剖析、经典错题解析以及强制性的手工代码训练，坚决杜绝此阶段内 AI 工具的介入；后续则切换为“现代范式融合场”，允许学生在受控环境下使用 Vibe Coding 进行逻辑重构验证，并借助 No-Code 工具绘制流程图或进行参数化界面的搭建。这种课内时间分割机制，根本上斩断了学生利用 AI 逃避核心思维训练的可能[6]。

以 C++面向过程部分的入门知识为例，设计如表 1 的教学设计表，充分显示了知识点、融合手段与思维培育目标的宏观映射关系，并全面展示了由浅入深的教学推演路径。

表2　四维协同评价打分表示例

协同评价维度	具体评价指标	权重	对应“星空案例”考核点	评估方式
计算思维（25%）	逻辑推理与算法设计	15%	能否设计“随机生成星星”“星星闪烁”的核心算法，用循环、随机数实现需求	实操编码+代码审核
	数据处理能力	10%	能否正确使用rand()生成随机坐标、大小，合理设置Sleep()延迟参数	实操编码+效果验证
工程思维（25%）	代码规范性	15%	drawStar函数注释、代码缩进、变量starCount命名是否合理，模块注释是否清晰	代码审核
	错误排查与优化	10%	能否排查“星空不显示”“闪烁异常”等错误，优化代码提升运行流畅度	实操测试+记录
No-Code思维与能力（30%）	需求拆解与可视化思维	8%	能否将“闪烁的星空”拆解为“创建画布、绘制星空、实现闪烁”3个模块	书面拆解+口头阐述
	组件化与复用思维	8%	能否复用drawStar函数绘制多颗星星，避免重复编写绘图代码	代码审核+实操验证
	直观化效果落地思维	7%	能否通过EasyX呈现黑色夜空、随机星星、闪烁效果，直观呈现需求	效果观察+学生反馈
	低门槛优化与交互思维	7%	能否通过修改starCount、Sleep()参数，调整星星数量和闪烁速度	实操优化+记录
协同落地效果（20%）	逻辑与效果协同	10%	编码逻辑（循环、随机数）正确，且能通过EasyX呈现对应的星空闪烁效果	实操验证+综合评估
	思维协同应用	10%	能否用编码逻辑实现星空绘制，用No-Code思维拆解需求、调整参数优化效果	口头阐述+实操记录

3．3 典型教学案例解析

纯粹的文本控制台输出常常令学生感到枯燥乏味，而过度庞大的第三方图形库框架（如 Qt 或 OpenGL）又会给初学者带来极其沉重的配置与学习负担。因此，本课程引入了 EasyX 这一轻量级图形库作为传统 C++代码与现代 No-Code 可视化思维之间的完美缓冲与中介。

以设计基于 EasyX 的“闪烁星空”项目为例，首先是基于 Vibe Coding 思维的纯逻辑需求拆解阶段。在未触碰任何代码之前，学生必须以产品经理的视角，利用白板或在线绘图工具将“动态星空”这一宏大需求解构为四大功能模块：底板画布的初始化机制、具备随机坐标/半径/颜色的星星生成引擎、包含清屏重绘延时逻辑的动画刷新模块，以及预留给用户的参数调整接口。这一过程深刻锻炼了计算思维中最关键的分解（Decomposition）能力。其次是 C++手工编码阶段，学生运用传统循环结构控制数十颗星星的批量渲染，如运用 rand()函数生成伪随机数，运用条件分支实现色彩的交替切换，并通过严格的函数封装（如 drawStar）实现代码级别的组件化。这一环节绝不容许 AI 的大规模介入，确保了代码硬实力的磨砺。再次是 Vibe Coding 思维的系统级渗透阶段。在代码跑通后，教师引导学生进行系统架构层面的思考。要求学生将控制星星数量、闪烁频率的关键数值从散落的代码中提取出来，统一放置在程序顶部的全局常量或配置结构体中。这使得学生顿悟：只需修改一处参数，即可瞬间改变整个星空的宏观视觉效果。这种“修改配置即改变应用”的体验，正是对 No-Code 底层哲学最为生动的诠释。最后是基于人机协同的扩展创新阶段。在完成了基础要求后，学生被允许解锁 AI 工具的全部能力。他们可以通过与 AI 对话，询问如何利用 C++算法让星星具备抛物线运动轨迹，或者如何加入键盘事件实现鼠标点击生成流星。此时的 AI 不再是越俎代庖的代笔者，而是激发学生进行创意拓展的高级技术顾问。这种从抽象逻辑到绚丽图形的即时视觉映射，极大激发了学生的编程内驱力。

3．4 对计算思维培育融合式教学的评估

在生成式 AI 工具唾手可得的今天，传统的仅依赖“期末闭卷笔试（考察语法死记硬背）”或“OJ 在线评测系统（仅看重最终输出的正确率）”的单维评价体系已难以适应新的形式变化。学生看似完美的代码作业，很可能是大模型的作品，而完全无法反映其计算思维的真实生长状况。有鉴于此，构建一套多维立体、重过程轻结果的协同评价量规，成为本课程改革的核心关键。

随着 AI 时代对布鲁姆教育目标分类学（Bloom's Taxonomy）的重新诠释，单纯的代码记忆与低阶实现已失去评估价值，必须全面扫描学生在分析、综合、评价与创造这些高阶认知维度的轨迹。这意味着，评估的焦点必须从冰冷的“代码成品”向活生生的“开发过程”转移——不仅要看代码写得如何，更要看设计思路是如何演进的，元认知是如何进行自我调节的。 基于这一哲学转向，本研究制定了总分为 100 分的“四维协同评价模型”，该量规的设计兼顾了传统硬核能力与现代前沿素养，具有极高的实操性，以对前例“闪烁星空”项目的评价为例，设计如表 2 的评分表格。

3.5 教学改革实施效果分析

为系统考察 Vibe Coding 融合教学模式的实施成效，本文采用平行班对比的准实验研究设计。研究对象为同一年级两个平行班共 82 名学生，其中实验班在常规 C++教学基础上引入 Vibe Coding 辅助学习和 EasyX 可视化项目实践，对照班则沿用传统程序设计教学方式。为保证比较结果的有效性，两班在教学内容、授课进度、学时安排和考核标准等方面保持基本一致。课程实施完成后，采用问卷调查与机考成绩相结合的方法开展效果评估：前者用于获取学生对课程体验及能力发展的主观感知，后者用于检验学生程序设计能力的客观达成情况。通过对两类数据的综合分析，可以较为全面地呈现融合教学模式的教学效果。

3.5.1 问卷内容简介与结果分析

问卷采用 5 点 Likert 量表，围绕学习兴趣与投入、计算思维、工程实践与调试能力、AI 协同与 Vibe Coding 认知四个维度设置题项。问卷内容重点考察学生在学习动机、问题拆解、程序调试、规范意识以及 AI 辅助使用等方面的主观感知，以反映融合教学模式对学生学习体验与能力发展的影响。

由表 3 可见，实验班在四个维度上的均值均高于对照班。其中，AI 协同与 Vibe Coding 认知维度差异最为明显，实验班均值为 4.35，对照班为 3.46，说明引入 Vibe Coding 辅助学习后，学生在人机协同、提示表达和 AI 生成结果判断等方面形成了更明确的认知。学习兴趣与投入维度上，实验班均值为 4.12，高于对照班的 3.78，表明可视化项目与 AI 辅助工具在一定程度上提升了学生参与编程学习的积极性。计算思维和工程实践维度上，实验班也分别高于对照班，说明该模式对问题拆解、流程设计、调试优化和代码规范意识具有积极促进作用。

3.5.2 机考成绩分析

为进一步检验教学改革对学生实际编程能力的影响，本文对两个班级的机考成绩进行统计分析。机考主要考察学生在限定时间内完成程序设计任务的能力，评价内容涵盖代码实现、逻辑完整性、调试排错和综合应用等方面。统计结果如表 4 所示。

表3　两班问卷结果分析表

维度	主要测量内容	实验班均值	对照班均值	分析结论
学习兴趣与投入	学习动机、参与意愿、持续投入	4.12	3.78	实验班学习兴趣与参与意愿更高
计算思维	问题拆解、流程设计、算法构思	4.05	3.81	实验班在问题拆解和逻辑建模方面表现更优
工程实践与调试能力	代码规范、调试排错、参数优化	3.98	3.85	实验班在工程实践能力方面略高于对照班
AI协同与Vibe Coding认知	提示表达、结果判断、合理使用AI	4.35	3.46	实验班在人机协同认知方面优势较明显

由表 4 可见，实验班平均分为 78.9 分，高于对照班的 75.2 分；实验班优秀率为 31.0%，高于对照班的 25.0%。两班及格率差异较小，实验班为 90.5%，对照班为 90.0%，说明两班学生整体均能达到课程基本要求。但从平均分和优秀率看，实验班在高水平学习成果形成方面表现更好。结合问卷结果可以认为，Vibe Coding 融合教学模式并未削弱学生的基础代码训练，反而在提升学习投入、促进问题拆解和增强综合编程表现方面具有一定积极作用。

表4　两班机考成绩分析表

班级	人数	平均分	标准差	及格率	优秀率
实验班	42	78.9	12.4	90.5%	31%
对照班	40	75.2	13.9	90.0%	25%

4 结论与展望

经过近一年的教学实践表明，本文所构建的融合教学模式成效显著，可有效降低程序设计入门难度，提升学生学习兴趣与参与度，强化计算思维与程序排错能力，培育学生现代工程素养。但实践中仍存在学生过度依赖 AI、教师工具学习与环境配置成本增加、评价难以兼顾个体差异等问题。但总的来说，本文设计的 C++与 Vibe Coding、无代码范式的融合的教学模式，可在不弱化学生代码能力的前提下提升教学效果，具备可复制、可推广价值，适用于高校 C++基础教学。后续我们将建设分专业差异化案例库、开发 AI 自动评测系统，推进跨学科融合与评价体系完善，助力程序设计教育高质量发展。

参 考 文 献

[1] 马昱春. 新质生产力发展背景下基于大模型的现代 C++教学模式设计[J]. 计算机教育, 2024(08).

[2] Horvat, Marko. (2025). What is Vibe coding and when should you use it (or not)?. 10.5281/zenodo.16747092.

[3] 邵晓艳，闫静静，李玲玲，等.C 程序设计课程思政元素挖掘与研究 [J]. 计算机技术与教育学报，2025,13 (4):93-98.

[4] 李双龙，徐舒婷，张娟. 国际计算思维教学研究现状与前沿动向分析[J]. 电化教育研究，2024(07).

[5] 郝熙，潘晟旻。面向 GenAI 的师 - 机 - 生三元共生程序设计教育模型研究与实证 [J]. 计算机技术与教育学报，2025,13 (3):18-23.

[6] 刘晋德，张江江，李诗珂。基于大模型的 C 语言课程教学改革探索与实践 [J]. 计算机技术与教育学报，2026,14 (1):60-65.