

Construction and Practice of an AI-Empowered Chain-Based Inquiry Teaching Model for C Programming

Huaizhao Wang

Southwest Petroleum University, Chengdu, Sichuan, 610500, China

ABSTRACT

C Programming is a foundational core course for computer-related majors in higher education. It has three cognitive barriers: intricate syntax, the opaque "black box" of compilation, and disconnection between teaching cases and practical application. Traditional teaching methods struggle to cultivate students' metacognitive abilities, while conventional inquiry-based teaching is constrained by two bottlenecks: difficulties in visualizing abstract concepts and high costs of constructing individualized learning pathways. To address these problems, this paper proposes an AI-empowered chain-based inquiry teaching model for C Programming and develops a locally deployed intelligent teaching platform. Taking the "inquiry task chain" as the main teaching thread and "diagnosis-probing-archiving" as the core teaching logic, the platform organizes learning processes through eight-step progressive inquiry activities. It provides dynamic learning scaffolds relying on heuristic probing of large language models and compilation behavior diagnosis, and realizes whole-process evaluation based on a three-dimensional model. A comparative experiment was carried out on two parallel classes in our university. The experimental class achieved significantly higher scores in final examinations than the control class and gained positive outcomes in the development of inquiry competency and learning experience. The results preliminarily verify the feasibility and effectiveness of the proposed model in improving academic performance and fostering students' autonomous inquiry habits.

KEYWORDS

Artificial intelligence; C Programming; Inquiry-based teaching; Process-oriented assessment

As a core foundational course for science and engineering majors, C Programming faces three typical cognitive barriers: intricate syntactic rules, the opaque "black box" of program compilation, and disconnection between teaching cases and practical application. Traditional lecturing and question-bank training fail to foster students' metacognitive competence, while inquiry-based teaching is constrained by two major bottlenecks: the difficulty of visualizing abstract concepts and the high cost of developing individualized learning paths^[1]. The superior capabilities of generative AI in code comprehension, real-time error diagnosis, and multi-round interaction perfectly match the demand for just-in-time scaffolding and personalized feedback in inquiry-based teaching. Nevertheless, existing research has not systematically embedded generative AI technologies into the full workflow of programming teaching^[2-3]. Accordingly, this paper proposes an AI-empowered chain-based inquiry teaching model for C Programming courses and develops a locally deployed intelligent teaching platform, whose effectiveness is verified via comparative experiments.

1 Research Value

As a core foundational course for science and engineering majors, C Programming undertakes the responsibility of cultivating students' computational thinking and cognitive competence of low-level system operations. However, its inherently high cognitive load frequently triggers learning anxiety, which further diminishes students' learning motivation and weakens their problem-solving capabilities. Traditional lecturing and question-bank training only strengthen students' mechanical syntactic memorization but fail to nurture their metacognitive competence. In this context, students tend to apply programming templates passively without conducting in-depth reflection on algorithm design, code debugging and error troubleshooting. Although inquiry-based teaching has been proven effective in stimulating students' intrinsic learning motivation, it faces prominent implementation barriers when deeply applied to C Programming instruction. Abstract concepts including pointers, recursion, and memory layout cannot be readily visualized during independent inquiry, resulting in a high trial-and-error cost for students. Furthermore, the design of individualized inquiry paths heavily relies on teachers' time and energy, making differentiated teaching impractical in large-class teaching scenarios.

The superior capabilities of generative AI in code comprehension, real-time error diagnosis, and multi-round interactive consultation achieve effective technological coupling with inquiry-based teaching, providing an innovative technical approach to breaking through the aforementioned teaching bottlenecks. On this basis, this paper constructs an AI-empowered chain-based inquiry teaching model for C Programming, whose core educational value is reflected in the following three dimensions:

(1) Reshaping classroom ecology to enable adaptive inquiry learning. The proposed model intelligently decomposes knowledge points and difficulty hierarchies of teaching cases and constructs a systematic inquiry task chain covering

foundational, advanced and innovative learning stages. Students with different academic proficiency can independently select appropriate task entries according to their own levels, complete program debugging, and summarize operational rules through progressive learning tasks. This practice fully implements the student-centered educational philosophy and substantially shifts classroom focus from teacher-dominated lecturing to student-oriented autonomous inquiry.

(2) Bridging cognitive gaps to realize visualized diagnosis of black-box program execution. Targeting the implicit characteristics of C language memory management and underlying program execution, the model integrates AI-based compilation diagnosis and whole-process learning data. It converts unobservable underlying logical information, such as variable addresses and data storage mechanisms, into intuitive visualized diagnostic reports and student cognitive profiles, effectively lowering the threshold for autonomous inquiry and learning.

(3) Upgrading the interactive paradigm to build a teacher-student-machine tripartite collaborative mechanism. AI undertakes procedural teaching work including code specification inspection and basic academic Q&A, freeing teachers from repetitive trivial work to concentrate on guiding students' high-order thinking and organizing discussions on algorithmic strategies. Meanwhile, full-dimensional learning behavioral data are continuously collected to support precise teaching intervention and dynamic instructional design. This mechanism aims to cultivate the human-machine collaborative programming literacy required by emerging engineering education, specifically the core competence to critically evaluate AI-generated suggestions and take the initiative to control algorithmic design ideas [4].

2 Design of the Intelligent Platform for Chain-Based Inquiry Cases

To realize the pedagogical principle of "student-led inquiry with AI-assisted guidance," the generative, diagnostic, and interactive capabilities of AI must be embedded in an orderly way into both case construction and the whole learning process. To this end, this study designs and implements a locally deployed intelligent teaching platform based on Ollama. Its core design idea is to use the "inquiry task chain" as the instructional logic backbone and a "diagnosis-probing-archiving" loop as the technical support closed loop. The following elaborates on the platform from three aspects: its overall architecture, the structured organization of case tasks, and the supporting mechanisms for the inquiry process.

2.1 Overall Platform Architecture

Positioned as an auxiliary tool for C Programming inquiry teaching, the platform adopts a locally hosted web-application architecture with separated front and back ends, keeping teaching data on local machines. The front end provides code editing, diagnosis visualization, and task-navigation interfaces; the back end handles case-data management, compilation diagnosis, and large-language-model inference services. The core engine invokes a locally deployed generative LLM to provide heuristic probing and uses a lightweight relational database to record the entire inquiry process.

2.2 Structured Organization of Case Tasks

To make the case chain extensible and reusable, the platform uses structured tables (in CSV format) to describe each inquiry node. Each node contains key information such as a case identifier, core knowledge point, task statement, expected evidence, heuristic probing focus, and difficulty level^[5]. Taking the "array out-of-bounds diagnosis" case as an example, Table 1 shows a structured-design example of three consecutive inquiry nodes.

Table 1 Structured design of array out-of-bounds inquiry nodes

task_statement	difficulty	prompt_focus
Read the following code: <code>int arr[5]; for(i=0; i<=5; i++) arr[i]=i;</code> Do you think the program will run correctly?	Basic	Consider that array indices start at 0; what is the valid range when the length is 5?
After changing the loop condition to <code>i<5</code> , the program runs normally. Explain why no unexpected values appear in the output?	Advanced	Why might out-of-bounds access not crash the program immediately? How might adjacent variables in memory be affected?
If the array length is changed to <code>n</code> (user input), how can the loop boundary be controlled dynamically? Try rewriting the code and verify.	Innovative	What is the difference in bounds checking between dynamic memory allocation and static arrays?

As Table 1 shows, the three nodes progress in their task statements from "observing phenomena" to "explaining principles" and then to "transfer design," exhibiting a stepwise rise in cognitive gradient. The difficulty level advances from "Basic" to "Innovative," and the probing focus moves from the syntactic level to the principle level and then to the design level—ensuring that the LLM, under these structural constraints, outputs heuristic questions with sound pedagogical logic. Teachers can extend cases autonomously following this field structure, enabling the case library to grow dynamically.

2.3 Technical Support Mechanisms for the Inquiry Process

The platform provides full-process technical support for inquiry-based teaching from three dimensions: chain-based task arrangement, heuristic guidance, intelligent diagnosis and archiving, as well as procedural evaluation.

(1) Arrangement of chain-based inquiry tasks. Based on the cyclic inquiry framework of “orientation-conceptual inquiry-empirical verification-reflection” and aligned with the cognitive rules of C language learning, the platform organizes all learning activities into an eight-step progressive chain: scenario reading, problem proposing, code observation, hypothesis verification, design modification, operation comparison, transfer practice, and conclusion induction. The chain supports iterative backtracking under systematic guidance to adapt to students’ differentiated cognitive rhythms. It reorganizes scattered exercises into a coherent and progressive cognitive path, enabling students to achieve gradual knowledge internalization.

(2) Heuristic probing guidance. The platform invokes locally deployed large language models to generate targeted heuristic probing at each task node. The probing strategy only provides directional thinking cues and hierarchical prompts rather than complete code solutions, which prevents students from skipping independent inquiry and ensures substantive knowledge exploration. Local deployment also guarantees data privacy and stable teaching service delivery.

(3) Diagnosis, archiving and procedural evaluation. The backend system is equipped with a dedicated diagnosis engine to automatically identify typical programming errors, including loop out-of-bounds, parameter transmission failures, and invalid pointer dereferencing. Visual diagnosis cards are generated to intuitively present program running states and students’ key knowledge deficiencies. All learning behavioral records, including code modifications, running results, diagnostic feedback, and reflective responses, are automatically archived in individual learning portfolios, forming a fully traceable evidence chain of inquiry practices. Three core indicators, namely task completion rate, code revision frequency, and interactive inquiry times, are adopted to dynamically portray the developmental trajectory of students’ inquiry competence. This strategy supports fine-grained procedural evaluation and achieves the teaching objective of promoting in-depth inquiry through whole-process assessment.

3 Practice of the Platform-Supported Chain-Based Inquiry Teaching

3.1 Teaching Implementation Framework

(1) Pre-class: Tiered guided learning based on case chains. Teachers select systematic case chains on the platform and establish three hierarchical teaching objectives covering basic, advanced, and innovative levels. The platform generates differentiated guided learning tasks according to difficulty gradients. The basic level provides conceptual discrimination exercises and standard code examples to consolidate foundational knowledge; the advanced level requires students to independently predict program running results and analyze potential errors; the innovative level designs life-oriented mini inquiry tasks to expand practical application capabilities. After students complete online self-assessment, the platform automatically aggregates and analyzes overall learning data, assisting teachers in determining the core teaching topics for subsequent classroom instruction.

(2) In-class: Collaborative inquiry empowered by heuristic probing. Teachers release project-based inquiry themes for classroom learning. Student groups first independently disassemble research ideas and sort out implementation logic. When encountering learning bottlenecks, students acquire hierarchical heuristic prompts from the large language model, and advance their learning following the standardized cognitive loop of “hypothesis-verification-correction-generalization”. Upon code submission, the system automatically detects typical programming faults, including pointer dereferencing anomalies and parameter transmission errors. Each group delivers on-site presentations, and all groups conduct peer evaluation and in-depth discussions with the support of visual teaching resources. Teachers provide targeted guidance for stagnant learning groups throughout the session without directly offering complete code solutions, ensuring authentic student inquiry.

(3) Post-class: Evidence chain archiving and tiered practical training. The platform generates personalized differentiated tasks based on real-time classroom learning data. Students with insufficient knowledge mastery receive targeted basic error-correction exercises, while students with solid foundational skills are assigned comprehensive practical projects for ability expansion. All coding behaviors and learning outcomes on the platform are automatically archived as multi-dimensional learning evidence in personal portfolios, forming a complete traceable inquiry record. Teachers summarize common learning problems through platform data analysis to conduct centralized Q&A and targeted tutoring in the next class.

(4) Whole-process inquiry-based evaluation. Replacing the traditional single summative assessment, the platform constructs a multi-dimensional weighted procedural evaluation model. It comprehensively records the full learning process including pre-class self-assessment performance, in-class code debugging behaviors, and post-class task completion status, and collects core evaluation indicators such as interactive inquiry frequency and code iteration revision times. Whole-process behavioral data serve as the core basis of assessment, with the final comprehensive programming project as a supplementary reference. Teachers ultimately determine students’ academic performance

through comprehensive evaluation of platform learning data, inquiry practice reports, and on-site defense performance.

3.2 Teaching Effectiveness

This study implemented the proposed teaching model among first-semester freshmen majoring in computer-related disciplines in 2025. A quasi-experimental design was adopted with two consecutive student cohorts under identical teaching conditions, including consistent instructors, class hour arrangements, textbooks, and assessment syllabi, ensuring the validity of the comparative analysis. The earlier cohort of 84 students was designated as the control group receiving traditional lecture-based instruction combined with routine computer lab training. The subsequent cohort of 90 students served as the experimental group receiving the platform-enabled chain-based inquiry teaching intervention. Both groups were evaluated under unified final assessment criteria.

(1) Academic performance comparison. Statistical results indicated substantial differences in comprehensive final scores between the two groups. The average score of the experimental group reached 82.7, while that of the control group was 74.5. In terms of grade distribution, 42.1% of students in the experimental group scored above 80 points, compared with 35.0% in the control group. Meanwhile, the failure rate of the experimental group was only 4.8%, considerably lower than the 12.5% recorded in the control group. These findings demonstrate that the proposed teaching model can effectively improve students' overall academic performance and reduce the proportion of academically underachieving students.

(2) Student learning experience and feedback. A post-semester teaching satisfaction survey was conducted to evaluate students' perceptual learning experience. The experimental group presented positive recognition of the optimized teaching mechanism, scoring 4.2 out of 5 on the item of "AI probing promotes in-depth thinking", 4.3 out of 5 on "reasonably progressive task difficulty design", and 4.0 out of 5 on "diagnostic feedback facilitates error localization and knowledge remediation". Typical student comments further validated the practical advantages of the model: "The connected in-class case sequences facilitate systematic knowledge absorption" and "This inquiry-oriented learning mode enables me to thoroughly understand the underlying operational principles of programming".

In conclusion, the platform-supported chain-based inquiry teaching model yields promising effects in boosting students' academic achievement, cultivating autonomous inquiry competence, and enriching learning experience. The results preliminarily validate the feasibility and effectiveness of the proposed model for C Programming teaching reform.

4 Conclusion

In conclusion, this study constructs an AI-empowered chain-based inquiry teaching model for C Programming and delivers a systematic demonstration from three dimensions: platform design, practical teaching implementation, and experimental validation. The empirical results verify that the proposed model can effectively improve students' academic performance, facilitate the development of autonomous inquiry competence, and optimize overall learning experiences. Nevertheless, this study still has several limitations. First, the experimental cycle is limited to a single semester with a relatively small sample size. Second, the current teaching case library only covers two core modules, namely arrays and pointers. Future research will extend the experimental duration, expand the participant scale, and enrich the coverage of the case library. Furthermore, the intelligent algorithms and interactive mechanisms of the platform will be continuously optimized to improve its adaptive teaching capability and better support personalized programming instruction.

About the Author

Huaizhao Wang, Master's degree holder, Assistant Laboratory Technician, specializing in image recognition, computer major teaching, and virtual simulation experiment development.

References

- [1] Wang D D. Analysis of the Current Situation of Computer Programming Education and Research on Teaching Reform Strategies[J]. Journal of Taizhou University, 2024, 46(6): 73-79.
- [2] Lin N, ZuoY, Lu T. Research on the Teaching Reform of C Language Programming Course Driven by Digital Intelligence Technology[J]. The Guide of Science & Education, 2024(19): 104-106.
- [3] Wang Y X, Xu W H, Yu H M, et al. Opportunities and Challenges Brought by Generative AI to C Language Programming Teaching[J]. Computer Education, 2024(8): 133-141+145.
- [4] Ahat A, Alimjan G. Teaching Reform of the C Programming Language Course Integrating OBE Concept and AI Assistance under the New Engineering Background[J]. Journal of Contemporary Educational Research, 2025, 9(7): 197-202.
- [5] Liu Y. Research on the Teaching Reform of the Case-guided C Language Programming Design Experiment Course Based on the OBE Theory[J]. Experimental Science and Technology, 2024, 22(3): 94-99.