Journal of Education and Educational Research
ISSN: 2957-9465 | Vol. 21, No. 2, 2026

Research on the Teaching Practice of AI Teaching Assistants in C Language Programming Courses

Zhanqi Wang, Huajiang Su *

Guangdong University of Science & Technology, Dongguan, Guangdong, 523000, China
*** Corresponding author:** Huajiang Su

Abstract: Under the traditional teaching model, C language programming has long faced practical pain points such as high entry barriers for beginners, delayed feedback on programming practice, insufficient supply of tiered teaching, and excessive teaching burden on frontline teachers. This study, with the core support of the human-machine collaborative education concept, explores the normalized implementation path of deeply embedding AI teaching assistants into the entire process of programming classes, and constructs a ternary collaborative teaching framework of "teacher guidance, AI teaching assistance, and student subjectivity". The study simultaneously sorts out practical issues existing in current AI teaching applications, such as superficial problem-solving, excessive dependence of students on technical tools, and insufficient adaptability to curriculum scenarios, and proposes targeted optimization strategies accordingly. Practice has shown that this teaching model can effectively compensate for the shortcomings of traditional classrooms, providing a practical reference for improving the intelligent teaching quality of computer programming courses in universities.

Keywords: Human-computer Collaboration; Hierarchical Teaching; AI Assistant; Programming; Smart Teaching.

1. Introduction

With the deep integration of generative artificial intelligence and educational informatization, intelligent teaching tools have become the core enabler for curriculum reform in universities, driving the transformation from traditional lecture-based teaching to a personalized and precise teaching paradigm featuring human-machine collaboration. The "Artificial Intelligence Innovation Action Plan for Higher Education Institutions" issued by the Ministry of Education explicitly requires the reconstruction of teaching processes, innovation of educational models, and systematic improvement of classroom teaching quality relying on artificial intelligence technology.

C Programming is a foundational course for science and engineering majors, playing a crucial role in cultivating students' computational thinking, logical reasoning, and engineering practice abilities. The teaching quality of this course directly impacts students' subsequent learning and the development of their innovative capabilities, making it a vital component of talent cultivation in higher education for STEM fields. Traditional programming courses suffer from issues such as monotonous formats and weak relevance, failing to meet the current societal demands for cultivating applied talents. Therefore, this paper begins with the teaching status of programming courses in universities and proposes an AI-assisted teaching model for programming courses. Through a year-long teaching practice, the application effects, existing problems, and improvement measures of this model are explored, aiming to provide references for the intelligent reform of programming courses.

2. Analysis of the Teaching Reality and Deep Contradictions of C Language Programming Courses in Universities

2.1. The Individual Learning Differences Among Students Span a Wide Range, making it Difficult for Traditional Stratified Teaching Schemes to Adapt to Large Class Teaching Scenarios, and the Implementation is Challenging

As an introductory core course shared by multiple science and engineering majors, C Language Programming has a wide range of students with varying levels of programming experience in the pre-requisite courses, resulting in distinct learning stratification characteristics. A small number of students with programming experience from junior and senior high school can quickly grasp core concepts such as variables, loops, and algorithms, and even complete extended programming tasks independently. However, the vast majority of students with no foundation generally face challenges in accepting, understanding, and mastering the unfamiliar programming logic, fragmented syntax rules, and counterintuitive program debugging logic. Under the traditional large-class unified teaching model, teachers can only design the teaching pace based on the average student level, failing to cater to the needs of both extremes: students with weak foundations fall further and further behind, with knowledge gaps accumulating, quickly losing confidence in the course; students with surplus learning abilities are "not fed enough", lacking suitable advanced learning resources. Students with surplus learning capacity feel "unsatisfied," lacking suitable materials for further study, gradually losing interest and enthusiasm in class, ultimately leading to a severe polarization in classroom learning outcomes, making the

concept of individualized instruction difficult to implement in general teaching activities [1].

2.2. After Completing Programming Practical Tasks, Students Often Fail to Receive Timely, Targeted Guidance, and the Accumulated Sense of Frustration is a Primary Cause of Student Attrition.

Practicality is the most crucial feature of programming courses. The process of having students write code, identify errors, and debug programs in class is how they master programming skills and develop an engineering mindset. However, in traditional large-class settings, when students encounter syntax errors, logical issues, or program malfunctions, they can only raise their hands and wait for the teacher to answer. A single instructor cannot promptly address every student's question when facing dozens or even hundreds of students, so many remain unanswered for tens or even dozens of minutes. Prolonged struggles with unresolved issues—constantly facing the pain of writing but failing to run programs—easily instill fear of programming. Students gradually become reluctant to learn, hesitant to engage, and lose motivation and confidence. This ultimately leads to high failure rates in the course. Although students may superficially grasp syntax knowledge, their practical skills remain poor, and the computational thinking they should develop in the course is inadequately cultivated.

2.3. Course Instructors Are Responsible for A Significant Amount of Non-Teaching Administrative Work, Which Severely Occupies the Time and Energy They Could Otherwise Dedicate to The Core Teaching Components of the Course

The complete teaching chain of the C language programming course covers multiple core aspects such as theoretical teaching, case demonstration, practical guidance, homework correction, error review, and learning analysis. The practical tasks throughout the entire process generally exhibit typical characteristics of heavy workload and strong repetition. Programming assignments are manually corrected, which is not only time-consuming and labor-intensive but also difficult to accurately summarize common errors at the class level and individualized problems of students. The efficiency and accuracy of the correction process are both at a low level. At the same time, students have a strong demand for answers during their independent learning after class, requiring teachers to invest a lot of time and energy in handling a large number of basic and repetitive questions. This part of transactional work significantly occupies teachers' time for core teaching tasks such as teaching design optimization, teaching method innovation, and differentiated instruction for excellence and improvement, directly restricting the continuous iterative improvement of classroom teaching quality.

2.4. The Current Curriculum Evaluation System Has a Single Dimension, Focusing Solely on Final Outcome Assessments, With A Lack of Coverage of Process-Oriented Educational Guidance Throughout the Entire Learning Process

The traditional C language curriculum evaluation system has always placed summative assessment at the absolute core, mostly adopting a single assessment model of "a small amount of usual score + final theory written examination + final centralized practical operation". All evaluations are based on the final exam paper submitted by students at the end of the semester, completely neglecting to include process-oriented elements that truly determine core programming competencies, such as code writing standardization, debugging logic thinking, daily practical practice quantity, phased progress, and independent debugging and exploration ability. Under this model, teachers cannot see the entire semester's trial and error process of writing code and the fluctuation in mastery of each knowledge point. When it comes to adjusting the teaching pace and providing targeted guidance to different students, they have no accurate process data to rely on. Instead, they can only make subjective judgments based on limited classroom observations and past experience. Precise teaching and personalized education are simply out of the question, and the full-process educational value that the curriculum should carry is severely weakened [2].

3. Design of a Full-Process Collaborative Teaching Architecture for AI Teaching Assistants in C Language Programming Classrooms

3.1. Intelligent Lesson Preparation and Proactive Learning Prediction based on Big Data of Learning Situation, Achieving Precise Targeted Positioning of Key and Difficult Points in Course Teaching

The pre class teaching process utilizes AI assistants for intelligent chemistry situation prediction and refined lesson preparation, solving problems such as excessive reliance on personal experience, lack of clear goals, and low work efficiency in the past lesson preparation process. AI teaching assistants can automatically obtain students' early learning situation and pre class preview test scores, use learning situation profiling technology to evaluate the basic knowledge mastery of the whole class, and accurately predict the possible shortcomings of students in the key content of this class, summarize the common problems in the whole class, and help teachers develop targeted layered teaching strategies. At the same time, AI teaching assistants can automatically generate different levels of student preview databases based on the teaching plan and expected teaching progress of the course: providing explanation videos and basic exercises suitable for students with poor foundations at their current level; For students with a good foundation, more challenging engineering examples and open-ended questions are provided, so that students at all levels can receive effective preview guidance. In addition, AI teaching assistants can also assist teachers in automatically selecting question banks,

updating courseware, and improving practical cases, saving teachers a lot of time and allowing them more time to think about how to organize classroom teaching and how to solve major problems in teaching.

3.2. Relying on AI Teaching Assistants for Intelligent Classroom Interaction and Real-Time Practical Guidance, Promote the Optimization of Practical Teaching Mode in Programming Courses

In class is the core scenario for AI teaching assistants to deeply empower program design teaching, and its core value points towards systematically solving the common pain points that have long existed in traditional classrooms, such as lagging response to practical Q&A, incomplete coverage of learning situations, and low efficiency of teacher-student interaction. In the theoretical teaching stage, AI assistants can synchronize with the pace of the teacher's teaching to update the core knowledge points in class, automatically generate structured knowledge mind maps, lightweight knowledge shorthand cards, and transform the originally abstract and obscure algorithm logic and program execution process into interactive dynamic visual demonstration cases, greatly reducing students' cognitive threshold and understanding difficulty of core knowledge points. In the practical training phase, a three-level collaborative coaching model is constructed with "student self debugging as the core, AI real-time learning as the support, and teacher precise guidance as the bottom line": after students complete code writing, they can rely on AI assistants to complete real-time dual dimensional intelligent detection of grammar errors and logical loopholes. The AI adopts heuristic and conversational guidance and answering strategies throughout the process, guiding students to independently locate and troubleshoot problems through layer by layer questioning, strictly preventing direct output of complete standard answers, and systematically cultivating students' independent debugging thinking and problem-solving abilities. In response to the complex practical difficulties that AI cannot automatically solve, as well as the common problems that appear in the classroom, the system will automatically classify and summarize them, and push them to the teacher's end in real time, where the teacher will carry out targeted centralized explanations and individual precise Q&A. At the same time, AI teaching assistants can provide real-time statistics on the accuracy, task completion rate, and typical error distribution of classroom practical exercises, dynamically update the real-time learning profile data of the entire class, and provide accurate process data support for teachers to adjust teaching pace and optimize teaching strategies in real time[3].

3.3. A Hierarchical Training System Based on Dynamic Profiling and A 24/7 Intelligent Question Answering Mechanism to Achieve Extended Coverage of Programming Teaching Scenarios at All Times

In the post class teaching stage, AI assistants are relied upon to achieve a comprehensive extension of the teaching scene, and a closed-loop education system covering the entire pre class, in class, and post class processes is constructed to systematically solve the common pain points of lagging response to Q&A, lack of gradient stratification in practice tasks, and lack of continuous tracking of learning throughout the entire cycle that have long existed in traditional post class teaching scenarios. AI teaching assistants can accurately divide students' ability levels based on real-time dynamic learning profile data generated in the classroom, and use learning clustering algorithms to push differentiated post class training tasks that are highly adapted to their ability starting points for students at different levels. Basic level students focus on consolidating core grammar rules and practicing basic programming at the introductory level to build a solid foundation of core subject abilities; Gradual students focus on medium-sized comprehensive case training and code logic optimization training to systematically enhance their engineering and practical abilities; Students at the promotion level rely on innovative practical projects and specialized exercises in subject competitions to carry out advanced expansion training, truly achieving multi-dimensional hierarchical training to improve and compensate for deficiencies. AI teaching assistants can provide students with 24/7 intelligent Q&A services, responding promptly to various problems encountered by students during their independent programming process after class, accurately identifying the deep root causes of code errors, and outputting step-by-step optimization guidance suggestions. Throughout the process, they avoid directly outputting complete standard answers and guide students to independently complete problem corrections and improve their abilities. At the same time, the system can automatically complete intelligent correction of all programming assignments, automatically collect typical mistakes, generate a personalized set of mistakes for each student, and output an overall analysis report of the class's learning situation, accurately identifying students' individual ability shortcomings and class stage common cognitive blind spots, providing comprehensive process data support for teachers to carry out targeted after-school tutoring and iterative optimization of subsequent course content[4].

3.4. Innovate a Multi-Dimensional Process Evaluation System Oriented Towards the Growth of Capabilities Throughout the Entire Cycle, And Promote the Comprehensive Improvement and Upgrading of the Curriculum Education System

Relying on the real-time big data collection capability of AI teaching assistants across all learning scenarios, a multi-dimensional comprehensive evaluation system featuring bidirectional deep integration of "full-cycle process evaluation and standardized summative evaluation" is established. This fundamentally addresses the structural flaws of traditional course evaluation models, which are often one-dimensional and lead to one-sided outcomes. AI teaching assistants can seamlessly collect multi-source behavioral data covering the entire learning journey of students, including core metrics such as the quality of pre-class preview task completion, depth of classroom interaction participation, accuracy of on-site practical tasks, time spent on single code debugging, iterative improvement effect on typical error questions, quality of post-class stratified training completion, and cumulative duration of independent inquiry learning. Through a learning situation portrait model, originally

difficult-to-quantify factors such as learning attitude, degree of knowledge internalization, engineering practical ability, and computational thinking development level are transformed into traceable and comparable quantitative evaluation results. Ultimately, the full-cycle process evaluation data is weighted with the summative scores from the final theoretical written examination and centralized practical assessment according to scientific weights, forming the final comprehensive score for students' courses. This completely avoids the issue of one-sided evaluation caused by single summative assessment. At the same time, the system automatically generates personalized chemistry analysis reports covering ability dimensions, growth trajectories, improvement directions, etc., and sends them to teachers and students, making it convenient for teachers to provide targeted one-on-one guidance, and also allowing students to understand their shortcomings and room for improvement, thus achieving the goal of "promoting teaching and learning through evaluation"[5].

4. The Implementation Path and Full Cycle Teaching Practice Promotion Process of Precise Teaching in Programming Design

4.1. Preparation for the Configuration and Implementation of Teaching Practice Tools

This teaching activity uses an intelligent programming AI teaching assistant system designed specifically for the educational environment of universities as the main support tool. The system has functions such as intelligent lesson preparation assistance, automatic intelligent code problem judgment, intelligent code vulnerability location, heuristic layered Q&A, comprehensive learning situation image analysis, personalized resource layered push, and recording of the entire learning process. It can well meet the teaching needs of programming courses in universities. Before the start of the activity, the project team provided operational guidance to all participating teachers and students in different roles: for teachers, their respective responsibilities and operational requirements in the teaching environment of human machine cooperation; For students, there are legal and reasonable behavioral norms on the platform, which systematically guide them to learn how to use AI to help answer questions, check whether the code meets requirements, and organize their own error notes. It is also clearly stipulated that students cannot directly copy all the code generated by AI to avoid negative learning attitudes and ensure the smooth progress of the entire activity.

4.2. Implementation Plan for Full-Process Intelligent Teaching Assistant

The pilot adaptation phase lasts from the first week to the fourth week of teaching, during which the main task is to integrate AI teaching assistant tools with current programming courses, forming a scalable human-machine collaborative teaching model. The teacher uses the AI assistant's pre class learning situation prediction function to analyze students' pre class learning situation, provide corresponding pre class learning materials for students at different levels, formulate rules for using AI tools in the classroom, and guide students in the practical process by adopting a "student-centered code debugging and AI error correction" approach. After class, the AI teaching assistant is used to automatically grade all programming assignments and summarize the learning situation of the entire class. A survey questionnaire is distributed or teacher-student communication is conducted to obtain the opinions and suggestions of the two main users on the AI teaching assistant. Based on these opinions, the accuracy of the AI teaching assistant's answers to questions and the degree of hierarchical resource matching are improved, thus forming a relatively mature teaching mode for AI teaching assistants to enter programming courses.

The time span of the comprehensive implementation phase is from the 5th week to the 20th week, during which the application of the intelligent teaching system in the four aspects of pre class, in class, post class, and evaluation is completed. In teaching activities, we strictly follow the standard process of layered and precise preview, intelligent classroom assistance, personalized training after class, and comprehensive and multi-dimensional evaluation. AI teaching assistants are responsible for completing tasks such as automatic homework correction, basic problem solving, and general student situation recording. This allows teachers to focus on solving major problems, cultivating students' computational thinking, carrying out engineering projects, and teaching students according to their aptitude. Throughout the entire process, all teaching processes and various learning activity information of students are saved, summarized and reflected on every teaching week, and the next teaching plan is adjusted based on the latest student situation, continuously leveraging the advantages of human-machine cooperation.

The summary and optimization phase lasts from the 21st to the 24th week of teaching. During this period, the main task is to collect data on the course learning achievements, programming proficiency tests, process learning attitude ratings, classroom performance, and other aspects of students in the experimental and control classes, as well as corresponding survey questionnaire feedback and semi-structured interviews with teachers and students. Based on this, inter group comparisons and process longitudinal tracking analysis are conducted to evaluate the effectiveness of AI assisted teaching in this project. The problems and shortcomings encountered during the entire project operation are summarized, and improvement measures are proposed to form a complete closed loop for the entire project.

5. Empirical Analysis of Teaching Practice Effects

5.1. The Practical Engineering Ability of Students in Programming is Improved Comprehensively and Systematically

Quantitatively comparing the core competencies of programming practice, it was found that the experimental class students with AI assistant assisted learning spent 25.9% less time on specific programming tasks assigned in class compared to the control class students without AI assistant assisted learning, and the proportion of first-time code submissions that passed in one go increased by 22.4% compared to the control class students; The experimental class students' ability to discover errors in the code and understand the internal structure of the program has greatly improved. Most students can independently complete difficult comprehensive programming problems or course design tasks. The quality of the code written and the

rationality of the program logic are significantly better than those of the control class students. However, due to excessive reliance on the help of teachers, students in the control class lack independent thinking when programming, resulting in common problems such as non-standard code writing, chaotic program logic, poor code fault tolerance, and poor problem-solving ability. In addition, students in the experimental class are more willing to participate in various programming related competitions or extracurricular innovation projects for training, and have a strong spirit of independent exploration and self-learning ability.

5.2. The Operational Efficiency of Classroom Teaching in Programming Courses Under the Human-Computer Collaborative Mode Has Achieved Significant Optimization Across the Entire Domain

Intelligent programming AI assistants comprehensively undertake repetitive teaching tasks with low added value, such as intelligent grading of programming assignments, automatic statistics of full-dimensional learning situations, instant response to high-frequency basic questions, and structured organization of teaching resources. According to calculations, this can reduce the total time spent on non-core transactional work by teachers by more than 50%, fully freeing up teachers' teaching energy from mechanical labor and promoting them to focus their core attention on high-level teaching links such as iterative optimization of course teaching design, systematic training of students' computational thinking system, in-depth decomposition and breakthrough of core knowledge difficulties, and individualized and precise education. At the same time, AI assistants can achieve millisecond-level instant response to fragmented questions generated during students' practical operations, eliminating the inherent pain points commonly found in traditional programming classroom manual answering, such as response lag, students waiting in line, and inability to reach all students with answering coverage. This significantly increases the effective teaching proportion per unit time in the classroom, reshapes the smooth running rhythm of classroom teaching, and ultimately achieves a significant positive improvement in the overall teaching operation efficiency of the course.

5.3. The Learning Attitude of Students in the Experimental Group Showed Significant Positive Optimization in the Dimension of Full-Cycle Learning Behavior

The cross-validation results of qualitative data from structured questionnaire surveys and semi-structured in-depth interviews with teachers and students show that 85.7% of students in the experimental group sample reported that the accompanying real-time tutoring provided by the AI assistant can effectively reduce the cognitive threshold of programming learning and significantly alleviate the learning frustration caused by prolonged stagnation during practical programming operations. 78.6% of students believed that the hierarchical learning resources pushed by the system were highly compatible with their current learning situations, and the precision and pertinence of the learning process were significantly enhanced. 78.4% of students indicated that the full-time question-answering response mechanism provided by the AI assistant could immediately solve various puzzles encountered during independent practical operations after class, and their intrinsic motivation and participation enthusiasm for autonomous learning were greatly enhanced. In sharp contrast to the common problems in the control group sample, such as fear of difficulty in programming learning, passive learning status, and learning weariness tendency among some students, the students in the experimental group showed significant positive improvements in classroom interaction participation, self-efficacy in programming learning, and independent problem-solving ability, ultimately forming a virtuous learning cycle of "low threshold guidance - positive feedback - ability progression".

6. A Targeted Optimization Strategy System for Teaching Practice Based on Empirical Data from the Entire Learning Cycle

The first improvement direction is the overall upgrade of the intelligent teaching technology base and school-based resource ecology, in order to better serve the entire teaching process. This measure is based on the teaching plan, main teaching objectives, and key and difficult content of school programming courses, and aims to transform the existing AI teaching resources into school-based ones, adding new content such as advanced algorithm learning, subject innovation project practice, and engineering comprehensive training; At the same time, improve the way and methods of AI Q&A, forming a "step-by-step questioning, inspiring thinking, and gradually correcting errors" model, fundamentally avoiding the situation where AI directly gives the correct answer, allowing students to think and draw conclusions on their own, thereby cultivating students' computational thinking ability.

The second aspect is the design of a standardized system for the entire process of AI teaching application, which aims to prevent students from engaging in superficial learning and academic misconduct. Specify the scope of AI tool usage in both classroom practice and self-study after class, and impose mandatory requirements for basic programming practice - students must complete the main part of the code writing work by themselves, and it is prohibited to directly use AI to generate all the code; And the originality testing of the code and students' ability to solve problems independently are included as assessment contents in the regular grades, which has a restraining effect on students' opportunistic behavior in the assessment, and helps students establish the correct concept of "tools are helpful rather than substitutes", using AI to broaden their learning path and improve their thinking level.

The third optimization path focuses on the dynamic optimization of role division in human-machine collaborative teaching, further strengthening the core leading value of teachers in the educational system. It always anchors on the core functional positioning of AI as "aiding teaching rather than dominating it, providing support rather than replacing education". Through technological undertaking, it completely separates teachers from low-value repetitive tasks in teaching activities, pushing teachers to focus their core energy on core aspects that intelligent technology cannot cover, such as deep decomposition of difficult and complex knowledge systems, systematic training of high-order computational thinking, full-process guidance of innovative projects, individualized and precise tutoring, value guidance, and emotional education. This fundamentally compensates for the natural shortcomings

of AI tools in the dimension of humanistic education, ultimately achieving a deep integration of technological tool empowerment and the goal of holistic education.

The final optimization direction is to design a dynamic closed-loop iterative mechanism around the entire lifecycle, so that human-machine collaborative teaching can continuously improve and develop itself. Establish a normalized multi-source data collection mechanism to obtain information on teachers' and students' experiences with using AI tools, as well as their entire learning process, at fixed time intervals as the main reference. Continuously optimize the matching degree of AI teaching resources with students' learning situations, accuracy in answering questions, and personalized learning resource recommendation methods; At the same time, according to the needs of curriculum and teaching reform, the content of practical training projects and process evaluation standards should be adjusted in a timely manner, thus forming a cyclical process of "teaching practice application - multi-source information aggregation - system upgrade and improvement - teaching effectiveness enhancement", ensuring that AI teaching assistants can flexibly adapt to various situations in daily classroom teaching.

7. Conclusion

Research has shown that there are still many shortcomings in the application of AI teaching assistants in teaching, such as simple problem solving, excessive student dependence, and insufficient educational role. These issues need to be addressed through measures such as technological upgrades, institutional improvements, and human-machine collaboration. In future teaching reforms, more emphasis will be placed on cultivating students' computational thinking, engineering skills, and innovation abilities. The connection between artificial intelligence and programming courses will be strengthened, and the human-machine collaborative education approach will be continuously improved. This will shift the curriculum from simply imparting knowledge to cultivating abilities and improving quality, thereby promoting the development of intelligent, personalized, and precise teaching in universities.

References

[1] Wang, L., Wang, T. M., & Shi, Y. J. (2026). Research on AI-empowered teaching reform of programming courses in higher vocational colleges. Cai Zhi, (29), 1–4. (in Chinese)

[2] Hu, L. S. (2025). Teaching reform and exploration of program-based courses based on large language models. Computer Knowledge and Technology, 21(30), 141–143. (in Chinese)

[3] Liu, P., & Chen, D. D. (2025). Research on AI-empowered reform of program design courses in higher vocational education. Electronic Components and Information Technology, 9(11), 278–280. (in Chinese)

[4] Luo, S. Z. (2025). Research on the teaching reform path of Python courses in higher vocational education based on generative artificial intelligence. Journal of Hubei Open Vocational College, 38(21), 158–160. (in Chinese)

[5] Peng, M. X., Zheng, F., & Wang, Y. (2025). Exploration of teaching reform in Java language programming course empowered by AI. University Education, (21), 58–63. (in Chinese)