Online ISSN: 3104-9338
Print ISSN: 3104-932X

Constructing a Generative AI Assistant for the Reform of University Experimental Teaching: A Case Study of the Advanced Language Programming (C Language) Course

Xinyu Song*

* School of Electronic Information Engineering, Geely University of China, ChengDu, China, 641423

Received: April 13, 2026

Revised: April 19, 2026

Accepted: April 22, 2026

Published online: April 27, 2026

To appear in: *International Journal of Advanced AI Applications,* Vol. 2, No. 5 (May 2026)

* Corresponding Author: Author Name (songxinyu@guc.edu.cn)

Abstract. With the rapid advancement of information technology, Advanced Language Programming (C Language) has become a core curriculum in computer-related disciplines in higher education institutions. However, the course's intrinsic characteristics—such as its strong practicality, rapid technological updates, and significant variance in student foundational knowledge—pose considerable challenges to traditional experimental teaching. To address these challenges and enhance instructional quality and personalized learning experience, this paper proposes and implements a Generative AI Teaching Assistant System (GenAI-TA) specifically designed for the "Advanced Language Programming (C Language)" experimental course. The system is built upon an advanced Large Language Model (LLM) and integrates Retrieval-Augmented Generation (RAG) technology. It is fine-tuned by incorporating a course-specific knowledge base (including the syllabus, lab manuals, code examples, and a collection of common errors) to provide precise, real-time, and personalized tutoring. This paper elaborates on the GenAI-TA's system architecture, key technical implementation details, and its integration scheme with the Geely University OJ Platform development environment. To evaluate its pedagogical effectiveness, a one-semester quasi-experimental study was conducted. The results indicate that, compared to the control group under the traditional teaching model, students in the experimental group using GenAI-TA achieved significant improvements in programming skills, project completion quality, and problem-solving abilities ($p < 0.05$). Furthermore, the System Usability Scale (SUS) score of 85.7 suggests a high level of student acceptance and satisfaction with the system. This research validates the immense potential of Generative AI teaching assistants in reforming the experimental instruction of practical courses and provides empirical evidence and a feasible technical pathway for the deeper application of AI in the higher education sector.

Keywords: *Generative Artificial Intelligence (GenAI), Advanced Language Programming, Large Language Model (LLM)*

1. Introduction

The course “Advanced Language Programming (C Language)” is an indispensable and crucial practical course in contemporary higher education programs such as Computer Science and Software Engineering. This course aims to equip students with the fundamental principles, methodologies, and core technologies of C language programming, enabling them to design and implement practical C language programs and lay a solid foundation for subsequent programming-related courses (such as Android or iOS development). The pedagogical core of the course lies in extensive programming experiments and project practice, which are vital for students to consolidate theoretical knowledge and cultivate engineering capabilities [1].

However, in the traditional experimental teaching model, the "Advanced Language Programming (C Language)" course faces numerous challenges. Firstly, the rapid iteration of programming technologies necessitates frequent updates to course content and lab projects, imposing extremely high demands on instructors' knowledge reserves and their capacity for developing teaching resources. Secondly, students exhibit significant differences in their programming foundation, logical thinking, and learning pace, making a uniform teaching schedule difficult to meet the personalized learning needs of all students. During the experimental process, students inevitably encounter various unexpected programming errors and environment configuration issues. Yet, the limited time and energy of instructors or traditional TAs make it challenging to provide 24/7, one-on-one, instantaneous feedback and guidance [2]. This lag in guidance often interrupts the student's learning flow (or "flow state"), dampens their enthusiasm, and can even lead some students to abandon complex lab assignments.

In recent years, Generative AI (GenAI) technology, represented by Large Language Models (LLMs), has achieved breakthrough progress and demonstrated immense potential for application in the education sector. GenAI's ability to comprehend natural language, generate code, explain complex concepts, and offer personalized, conversational interaction makes it an ideal tool for assisting experimental teaching and addressing the aforementioned challenges. By developing an AI teaching assistant specifically tailored to a particular course, it can provide students with a 24/7 available "expert partner" that instantly answers questions, debugs code, and offers learning suggestions, thereby achieving large-scale personalized education.

Although existing research has explored the application of AI in programming education and as a general instructional tutoring tool, studies focusing specifically on the systematic design, implementation, and empirical evaluation of an AI solution for a highly practical university lab

course like “Advanced Language Programming (C Language)” remain scarce. Specifically, there is a lack of cases that are deployed in a real classroom environment and measure its impact on student learning outcomes, engagement, and satisfaction [3-5].

In view of this, this paper takes the “Advanced Language Programming (C Language)” course as a case study, aiming to design, develop, and evaluate a Generative AI Teaching Assistant system (GenAI-TA) for the reform of experimental teaching in higher education. The main contributions of this research include:

(1) Proposing a Generative AI Teaching Assistant system architecture that integrates a course-specific private knowledge base, utilizing Retrieval-Augmented Generation (RAG) technology to ensure the accuracy and relevance of responses.

(2) Implementing the TA system and seamlessly integrating it as a plugin into the mainstream Geely University OJ Platform Integrated Development Environment, providing immersive learning support to students.

(3) Conducting a semester-long quasi-experimental study to quantitatively and qualitatively evaluate the pedagogical effectiveness of GenAI-TA from multiple dimensions, including learning gain, system usability, and student satisfaction.

The structure of this paper is organized as follows: Section 2 reviews related work; Section 3 details the system architecture and implementation of GenAI-TA; Section 4 describes the experimental design, data collection methods, and results; Section 5 provides an in-depth discussion and analysis of the experimental results; finally, Section 6 concludes the paper and outlines future work.

2. Related Work

This study is built upon two main foundational areas: the instructional reform of Advanced Language Programming (C Language) courses, and the application of Generative AI in the education sector.

2.1. Instructional Status Quo and Reform Exploration of Advanced Language Programming (C Language) Courses

The syllabus for the "Advanced Language Programming (C Language)" course typically covers several modules, including programming environment fundamentals, data structure application, file operation, pointer usage, function design, and basic algorithm implementation. In terms of pedagogical methods, there is a general emphasis on "learning by doing," utilizing

approaches such as case studies, Project-Based Learning (PBL), and the Flipped Classroom model to strengthen students' hands-on practical abilities. The experimental component is the core focus of the course, with lab types gradually progressing from basic verification and design-oriented tasks to comprehensive and innovative projects. For instance, students are required to complete a series of tasks ranging from simple ones, like "Basic Calculator Program Development", to more complex ones, such as "Student Information Management System Based on File Storage", and then "Simple Data Sorting and Searching Algorithm Implementation."[6]

To address the shortcomings of traditional instruction, numerous reform efforts have been explored. For example, some studies propose constructing a multi-level, modular laboratory system to meet the needs of students at different proficiency levels. Other research has explored cloud-based online lab environments to solve the difficulties students face with local environment setup. However, these reforms primarily focus on course content organization and pedagogical models. They still lack effective technical support tools for the instantaneous and personalized problems students encounter during the experimental process. Students' learning experience largely remains dependent on the on-site guidance of the instructor, which is often inadequate in the context of large-scale teaching.[7]

2.2. Application of Generative AI in the Education Sector

The emergence of Generative AI has brought about a new paradigm for educational innovation. In recent years (2022–2025), a large volume of research has emerged, exploring the role of AI as a learning partner, tutor, or teaching assistant. These studies generally agree that Generative AI can provide personalized feedback, stimulate students' learning interest, and enhance learning efficiency [8,9].

In terms of evaluation, researchers employ various metrics to gauge the effectiveness of AI TAs. For example, a 2025 study used an experimental design to assess the impact of Generative AI on student learning satisfaction, self-efficacy, and learning outcomes, finding that personalized AI support significantly enhanced these indicators. Other studies have also focused on student engagement, the usability of the AI tutor, and student acceptance. Regarding evaluation tools, the System Usability Scale (SUS) is widely used to measure users' subjective perception of an AI system's usability [10].

In the field of Computer Science education, AI code generators (such as GitHub Copilot) and LLM-based chatbots (such as ChatGPT) have been utilized to assist in programming learning. Research indicates that these tools can help students write code and debug errors more quickly.

However, concerns also exist, such as the potential for students to develop over-reliance, thereby weakening their ability to solve problems independently, and the possibility of the model generating insecure or inefficient code.

2.3. Research Gap

In summary, despite the increasing number of studies on the application of Generative AI in education, several research gaps persist: most existing research focuses on general educational scenarios or pure theoretical programming instruction, with highly limited research on AI Teaching Assistant systems specifically designed for highly practical lab courses like "Advanced Language Programming (C Language)"—which involves complex IDEs, data structure application, file operation, and pointer-based programming[11]; many AI tutoring tools exist as independent web pages or chatbot applications rather than being deeply integrated with the professional development tools (e.g., Geely University OJ Platform) that students use daily, disrupting the continuity of the learning process [12]; and although some preliminary user experience studies exist, long-term (e.g., a whole semester), controlled, multi-dimensional quantitative empirical evaluations—including metrics such as learning gain, project quality, and SUS scores—in real university lab courses are still rare.

To bridge these gaps, this study develops a GenAI-TA that is deeply integrated into Geely University OJ Platform and features a highly customized knowledge base, and conducts a comprehensive empirical evaluation of its effectiveness in a genuine instructional environment.

3. System Architecture and Implementation

To provide precise and efficient tutoring for the "Advanced Language Programming (C Language)" course, the GenAI-TA system we designed must possess the following core capabilities: (1) Deep comprehension of course-specific knowledge; (2) Accurate answering of factual questions and explanation of complex concepts; (3) Generation of high-quality example code conforming to course standards; (4) Understanding and assisting in the debugging of student errors; and (5) Providing a seamless interaction experience.

3.1. System Overview Architecture

Presentation Layer: This is the front-end interface for user interaction with the system. We designed it as a Geely University OJ Platform Plugin. The plugin provides a chat window in the IDE's sidebar, allowing students to ask questions using natural language directly within their C language programming environment without switching applications. The UI design is concise,

supports C language code block highlighting, one-click copy, and insertion functionality, ensuring a smooth user experience.

Application Layer: As the system's backend service, it is responsible for handling requests from the front-end, coordinating business logic, and calling the underlying models. It includes three core modules: Session Manager: Responsible for maintaining the conversation history of each user, providing context support for multi-turn dialogue. Intent Recognition Module: Conducts preliminary analysis of user input to determine the intent, categorizing it as "Conceptual Question" (e.g., pointer usage, file operation principles), "Code Request" (e.g., C language function implementation, loop structure writing), "Error Debugging" (e.g., compilation errors, logical errors in C code), or "Lab Guidance" (e.g., C language lab task completion tips).

The Query Processor constructs the appropriate Prompt based on the recognized intent and decides whether to directly call the LLM or first go through the RAG process. The Model & Data Layer serves as the intelligent core of the system, consisting of the Large Language Model (LLM) and the course-specific knowledge base; we selected an advanced open-source LLM (such as Llama 3 70B) as the foundation model, and choosing an open-source model facilitates localized deployment and deep fine-tuning, thereby better protecting student data privacy and controlling costs.

The Course Knowledge Base is crucial for ensuring that the AI Teaching Assistant's responses are professional and accurate, so we built a comprehensive, structured knowledge base for the "Advanced Language Programming (C Language)" course, with data sources including course instructional materials (course outline, instructor PPTs, text transcripts of official instructional videos focusing on C language core knowledge like data types, pointers, functions, and file operations), lab guidance documentation (objectives, steps, requirements, and grading criteria for all C language labs such as basic syntax practice, function development, and file operation experiments), code assets (all C language example code, project templates like student information management system and simple calculator program, and best practice samples provided by the instructor that adhere to C language coding standards), and FAQ & Error Collection (accumulated frequently asked questions from previous years such as pointer operation confusion and file opening/closing errors, typical C language compilation and logical errors and their corresponding solutions). All documents in the knowledge base are chunked, cleaned, and converted into high-dimensional vectors using a text embedding model (such as m3e-base), then stored in a vector database (such as ChromaDB), which enables quick retrieval

of C language-specific knowledge to assist the LLM in generating accurate responses.

4. Experimental Design and Results

To evaluate the effectiveness of the GenAI-TA system in a real instructional environment, we conducted a 16-week quasi-experimental study.

4.1. Participants and Grouping

The participants in this study were 98 undergraduate students enrolled in the “Advanced Language Programming (C Language)” course during the first semester of the 2024-2025 academic year at our university. These students were randomly assigned to two parallel experimental classes:

Experimental Group (N=49): Students were permitted to use the GenAI-TA system, integrated into Geely University OJ Platform, during lab sessions. They could also access the system for assistance outside of class hours.

Control Group (N=49): Students adopted the traditional learning approach. Guidance during lab sessions was provided by a human Teaching Assistant (a postgraduate student), and post-class questions were addressed via forums or email to the instructor or TA.

Both groups utilized the same syllabus, textbooks, lab assignments, and grading criteria. To ensure fairness, the query time for the human TA was restricted to the fixed lab hours. Before the experiment began, we conducted a t-test on the students' programming foundations (based on grades from the prerequisite course "Java Programming") and learning motivation questionnaire. The results showed no significant difference between the two groups ($p > 0.1$), indicating that the grouping was balanced.

4.2. Experimental Tasks and Data Collection

The course comprised 8 lab assignments, ranging from simple C language syntax practice to complex file operation and data structure application tasks, and the final major assignment required students to complete a comprehensive C language program independently or in small groups (maximum 2 people). We collected both quantitative and qualitative data through various means, including lab report scores, final project scores, system usage logs, the System Usability Scale (SUS), satisfaction questionnaires, and semi-structured interviews.

Specifically, we recorded the average score of the eight lab reports for each student, with grading criteria covering functionality implementation, C language code quality, and reporting standards across multiple dimensions; three instructors independently scored the final projects

based on completeness, innovativeness, technical difficulty, and code standardization, using the average score. For the experimental group, we recorded anonymized data on each student's interaction frequency with GenAI-TA, session duration, and question types (e.g., conceptual questions about pointers, code debugging, file operation guidance), while for the control group, we recorded the number of times students asked questions to the human TA during lab sessions. Additionally, students in the experimental group completed a 10-item SUS questionnaire (scoring 0 to 100) at the end of the semester to evaluate GenAI-TA's usability, both groups completed a 5-point Likert scale satisfaction questionnaire on their course learning experience and TA support effectiveness, and eight students were randomly selected from each group for in-depth interviews to understand their perceptions of the TA (AI or human), usage experience, and impact on their C language learning process.

4.3. Experimental Results

As shown in Table I, students in the experimental group achieved significantly higher scores in both the average lab report score and the final project score compared to the control group. Independent samples t-tests revealed that this difference was statistically significant ($p < 0.05$), indicating a positive impact of GenAI-TA's introduction on enhancing students' C language practical abilities and project completion quality.

Table 1. Comparison of practical performance metrics between the experimental group and the control group.

Evaluation Metri	Experimental Group (N=49)	Control Group (N=49)	t-value	p-value
Average Lab Report Score	89.5±5.2	84.1±6.8	4.312	0.001
Final Project Score	91.2±4.9	86.7±6.1	3.987	0.003

System logs indicated that students in the experimental group interacted with GenAI-TA an average of 15.3 times per week, which is significantly higher than the average number of questions asked by control group students to the human TA (approximately 2.1 times per week). Notably, the usage frequency of GenAI-TA increased significantly during non-working hours (evenings and weekends), accounting for over 65% of total interactions. This demonstrates that the AI Teaching Assistant greatly satisfied the students' need for "anytime, anywhere" C language learning support.

The average SUS score for GenAI-TA in the experimental group was 85.7 (SD=7.9), which is generally considered an "Excellent" level of usability. GenAI-TA System Usability (SUS) Score Distribution. In the satisfaction questionnaire, students in the experimental group rated

the "Timeliness of TA support" (average score 4.8/5.0) and the "Effectiveness of TA answers" (average score 4.6/5.0) significantly higher than the control group (which scored 3.2 and 4.1, respectively).

5. Discussion and Analysis

The experimental results of this study strongly confirm the positive role of GenAI-TA in the experimental instruction of the “Advanced Language Programming (C Language)” course.

Firstly, GenAI-TA significantly enhanced students' learning outcomes. The improvement in the experimental group's scores can be attributed to the following points: (1) Instant Feedback: Students could receive immediate answers when encountering C language bugs (e.g., compilation errors, logical errors) or conceptual queries (e.g., pointer usage, file operation principles), preventing prolonged periods of stagnation and frustration, thus maintaining learning continuity. (2) Deeper Exploration: The AI teaching assistant could provide relevant extended knowledge (e.g., advanced pointer applications, efficient file operation methods) and C language code examples based on students' questions, encouraging students to engage in more in-depth, exploratory learning. (3) Code Quality Improvement: GenAI-TA not only points out C language code errors but also explains the reasons behind them (e.g., syntax non-compliance, logical flaws) and offers suggestions for modifications that follow C language best practices, subtly fostering good programming habits in students.

Secondly, GenAI-TA greatly increased students' learning engagement and autonomy. The 24/7 availability feature broke the constraints of time and space, shifting learning behavior from being "concentrated in class" to "happening anytime, anywhere." In the interviews, one student from the experimental group mentioned: "Before, I had to wait until the next day's lab session to ask the teacher if I ran into a C language problem. Now, if I get stuck while coding C language programs late at night, I can ask the AI TA and get ideas within seconds. That feeling is fantastic." This indicates that GenAI-TA effectively supported students' personalized learning pace and promoted the cultivation of self-regulated learning ability in C language programming.

Thirdly, high usability and satisfaction are key to the successful application of the AI teaching assistant. The high SUS score of 85.7 confirms the success of our strategy to integrate GenAI-TA as a Geely University OJ Platform plugin. Students did not have to leave their familiar C language development environment, making AI assistance "readily accessible." In the interviews, students generally praised its conversational interaction style, feeling that it was "much more efficient than searching for fragmented C language answers online." This validates

the importance of good human-computer interaction design for the successful implementation of technology in education.

However, this study also uncovered some issues warranting attention. Some students mentioned in the interviews that they sometimes "subconsciously" ask the AI for complete C language code directly, rather than thinking through the problem first. This suggests that preventing student over-reliance is a crucial optimization direction for future work. Furthermore, although RAG technology significantly improved the accuracy of answers, GenAI-TA's responses could sometimes be generic when dealing with highly unconventional or innovative C language problems involving the complex interworking of multiple modules (e.g., integrating file operations with pointer applications), lacking profound insight.

The limitations of this study are as follows: (1) The experiment was limited to a single course ("Advanced Language Programming (C Language)") and a single institution, and the generalizability of its conclusions requires further verification. (2) The long-term impact of using GenAI-TA on students' independent C language problem-solving skills and creative thinking needs to be revealed through a longer-term longitudinal study.

6. Conclusion and Future Work

This study addressed the instructional pain points of the "Advanced Language Programming (C Language)" experimental course by designing, implementing, and empirically evaluating a Generative AI Teaching Assistant system named GenAI-TA. The research findings demonstrate that the system, by providing instant and personalized tutoring for C language programming, can significantly enhance students' learning outcomes, engagement, and satisfaction. This validates the immense potential of Generative AI in reforming the instruction of engineering practice-oriented courses, especially programming courses like "Advanced Language Programming (C Language)". This research provides a successful exemplar and a feasible technical solution for the deep integration of AI technology with specialized programming course experimental instruction.

Future work will focus on the following aspects:

(1) Intelligent Pedagogical Intervention: Incorporate a learner model into GenAI-TA to actively identify students' C language knowledge gaps (e.g., deficiencies in pointer operation, file handling, or data structure application) by analyzing their questioning patterns and C language coding behaviors. The system will then push personalized C language learning resources (e.g., targeted practice questions, code examples) and exercises, enabling a shift from

"passive Q&A" to "active guidance."

(2) Optimizing Interaction and Anti-Dependence Mechanism: Design a "Socratic" or "heuristic" answering mode. When the system detects potential student over-reliance (e.g., directly requesting complete C language code without independent thinking), it will prioritize offering conceptual hints and guiding questions rather than directly providing answers. This aims to cultivate students' critical thinking and independent C language problem-solving abilities.

(3) Generalization and Expansion: Generalize the GenAI-TA framework so that it can be rapidly adapted to other highly practical programming courses (such as "Web Development," "Machine Learning," etc.) by simply loading different course knowledge bases, thereby building a scalable AI Teaching Assistant platform.

(4) Conducting Larger-Scale Longitudinal Studies: Promote the application of the system in more universities and programming courses, and conduct long-term tracking research spanning several years to fully assess the long-term impact of Generative AI on students' C language professional competencies and lifelong learning capabilities.

We believe that with the continuous maturity of technology and the persistent optimization of pedagogical design, Generative AI will become a crucial engine driving the personalized, intelligent, and efficient development of higher education, particularly in the field of programming instruction.

References

[1] Krusche, S., & Alperowitz, L. (2018). Introduction of continuous delivery in multi-customer project courses. *In Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training* (pp. 31–40). IEEE. https://doi.org/10.1145/3183377.3183378

[2] Villegas-Ch, W., Román-Cañizares, M., & Palacios-Pacheco, X. (2020). Improvement of an education online model with the integration of machine learning and data analysis in an institution of higher education. *Sensors*, 20(5), Article 1396. https://doi.org/10.3390/s20051396

[3] Chiu, T. K. F. (2023). The impact of generative AI (GenAI) on practices of higher education: Challenges and opportunities. *Smart Learning Environments*, 10, Article 39. https://doi.org/10.1186/s40561-023-00258-w

[4] Kilde-Westberg, S., & Bøe, M. V. (2025). Generative AI as a lab partner: A case study in a university physics course. *Computers and Education: Artificial Intelligence*, 8, Article 100344. https://doi.org/10.1016/j.caeai.2024.100344

[5] Lau, S., & Guo, P. J. (2023). Banter: A pedagogically-appropriate AI tutor for computer science education. *In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM. https://doi.org/10.1145/3544548.3581564

[6] Essel, H. B., Vlachopoulos, D., Tachie-Menson, A., Johnson, E. E., & Baah, P. K. (2024).

The impact of generative AI (ChatGPT) on university students' engagement and learning gains: A systematic review. *Higher Education Quarterly*. Advance online publication. https://doi.org/10.1111/hequ.12504

[7] Brooke, J. (1996). SUS: A "quick and dirty" usability scale. *In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), Usability evaluation in industry* (pp. 189–194). Taylor & Francis.

[8] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *In Advances in Neural Information Processing Systems 33* (pp. 9459–9474). Curran Associates, Inc.

[9] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv*:2106.09685.

[10] Barke, S., James, M. B., & Polikarpova, N. (2023). Grounded copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA2), 85–111. https://doi.org/10.1145/3618307

[11] Holmes, W., Persson, J., Chounta, I. R., Wasson, B., & Dimitrova, V. (2022). Ethics of artificial intelligence in education: Towards a community-wide framework. *International Journal of Artificial Intelligence in Education*, 32(3), 504–526. https://doi.org/10.1007/s40593-022-00292-9

[12] Farrokhnia, M., Konijn, E. A., Akyüz, N., & Beer, N. (2024). A favourable buddy or a fearful beast? A systematic review of ChatGPT's roles, benefits, and challenges in education. *Educational Technology Research and Development*, 72(1), 1–43. https://doi.org/10.1007/s11423-023-10298-w