Journal of Contemporary Educational Research, 2025, Volume 9, Issue 5
http://ojs.bbwpublisher.com/index.php/JCER
ISSN Online: 2208-8474
ISSN Print: 2208-8466

Adapting High-Level Language Programming (C Language) Education in the Era of Large Language Models

Baokai Zu, Hongyuan Wang*, Hongli Chen, Yafang Li

College of Computer Science, Beijing University of Technology, Beijing 100124, China

Corresponding author: Hongyuan Wang, wanghongyuan@bjut.edu.cn

Abstract: With the widespread application of large language models (LLMs) in natural language processing and code generation, traditional High-Level Language Programming courses are facing unprecedented challenges and opportunities. As a core programming language for computer science majors, C language remains irreplaceable due to its foundational nature and engineering adaptability. This paper, based on the rapid development of large model technologies, proposes a systematic reform design for C language teaching, focusing on teaching objectives, content structure, teaching methods, and evaluation systems. The article suggests a teaching framework centered on "human-computer collaborative programming," integrating prompt training, AI-assisted debugging, and code generation analysis, aiming to enhance students' problem modeling ability, programming expression skills, and AI collaboration literacy.

Keywords: Large language models (LLMs); High-level language programming; C language; Human-computer collaborative programming

Online publication: June 4, 2025

1. Introduction

The High-Level Language Programming course is a key foundational course for computer science majors, responsible for the critical tasks of introducing programming thinking, training computational thinking, and developing initial engineering skills. As the core teaching language of this course, C language plays an irreplaceable fundamental role in the programming education system due to its concise syntax, rigorous structure, and proximity to low-level implementation. It has extensive applications in fields such as embedded development, system software construction, and data structure implementation.

In recent years, with the rapid development of artificial intelligence, particularly large language models (LLMs), intelligent tools represented by ChatGPT [1], GitHub Copilot [2], and others have demonstrated remarkable capabilities in text generation, semantic understanding, code generation, and debugging assistance.

Students are now able to interact with models through natural language, quickly generating well-structured, runnable C language programs [3,4]. This trend is profoundly reshaping student learning behavior, teaching strategies, and the content structure of the course itself [5].

However, most C language courses in current universities still follow the traditional teaching model of "teacher instruction + code practice," which has limitations such as closed content, delayed feedback mechanisms, and one-dimensional interaction methods, making it difficult to meet students' expectations for personalized, intelligent learning experiences in the modern era. At the same time, despite the strong assistive capabilities of large models, the challenge of how students can scientifically understand and use these tools, how to avoid over-reliance on technology, and how to enhance collaborative programming abilities has become an urgent issue in the teaching process [6–8].

In this context, constructing a new paradigm for the High-Level Language Programming course that integrates the capabilities of large language models and focuses on "human-computer collaborative programming" is not only an upgrade and optimization of existing teaching content and methods, but also a key path to cultivate students' engineering practice abilities and AI literacy in the future. Based on the current teaching status of C language courses in universities and the evolution trends of technology, this paper proposes a systematic teaching reform plan, exploring aspects such as the restructuring of teaching goals, content integration, teaching methods design, and the innovation of evaluation mechanisms, providing theoretical references and methodological support for the future implementation of teaching reforms.

2. Course background and reform opportunities

2.1. Course positioning and structure

High-Level Language Programming is a foundational course for computer science majors, typically taught in the first semester. It uses C language to help students understand program mechanisms and memory control principles, laying a foundation for subsequent courses like Data Structures and Operating Systems. The course is organized in four stages: basic syntax, control flow, data structures, and project practice. It emphasizes both technical skills and early development of programming thinking and engineering awareness.

2.2. Challenges in current teaching

Despite its structured approach, the traditional teaching model faces limitations in the context of AI and modern software development. The content is heavily focused on syntax and logic, with little integration of AI tools or modern engineering practices, making it difficult for students to connect theory with real-world applications. Furthermore, the learning process tends to be closed and imitative, with limited student exploration. Practical tasks are isolated, and there is insufficient feedback or guidance in debugging, affecting the development of problem-solving skills.

2.3. Opportunities for teaching reform with LLMs

The rise of LLMs offers new opportunities for reforming programming courses. LLMs, with their advanced language understanding and code generation abilities, can enhance students' learning and problem-solving skills. In High-Level Language Programming teaching, LLMs can assist in generating code and providing real-time feedback during debugging. Prompt engineering offers dual training in language expression and problem modeling, enabling students to transition from understanding problems to coding solutions. This shift from "knowledge transfer" to "ability generation" can transform the course into a task-driven, human-AI

collaborative learning environment.

3. Teaching transformation pathways under AI empowerment

3.1. Shift in teaching philosophy and goal orientation

With the rapid development of LLMs, there is an urgent need for the transformation of C language teaching from a "syntax training-oriented" model to a "collaborative skills-oriented" approach. This reform centers on the core philosophy of "AI-assisted + problem-driven + critical collaboration," emphasizing the systematic development of students' expression, validation, and iteration abilities in the context of technological empowerment. The reform follows the fundamental principles of "balancing knowledge and skills, integrating theory and practice, and ensuring human-machine collaboration," aiming to establish a new teaching system that is both in line with beginners' cognitive development and oriented toward engineering practice.

To effectively implement this philosophy, the teaching objectives are broken down into five interrelated ability dimensions. First is language foundation ability, requiring students to master C language syntax, logical expressions, and modular design methods. Second is problem modeling ability, enabling students to abstract real-world problems into programming tasks and clearly express requirements using natural language. Third is prompt engineering ability, where students should be able to write high-quality prompts to guide AI models in generating structured and semantically accurate code. Fourth is AI collaboration and optimization ability, which includes evaluating, modifying, and debugging AI-generated results, fostering a collaborative mindset between humans and machines. Finally, engineering awareness and reflective ability are cultivated through project practice, focusing on code standards, development processes, and teamwork, with reflection aimed at enhancing programming literacy and problem-solving skills.

This multidimensional goal system not only covers the traditional focus on foundational knowledge but also introduces future-oriented collaborative intelligence literacy, creating a new teaching direction from "mastering tools" to "driving tools."

3.2. Curriculum design and methodological innovation

Guided by the aforementioned ability framework, the course content design follows the principles of "solid foundation, clear progression, task-driven, and AI integration," forming a four-level progressive module system: "basic syntax training—structural design deepening—AI collaboration introduction—engineering practice expansion." In the initial phase, the course focuses on basic syntax elements such as variable definitions, data types, control structures, arrays, and functions, using simple cases and step-by-step exercises to help students build language foundations. In the middle phase, modular programming methods, debugging skills, and more complex structures like pointers and structs are introduced, enhancing students' logical thinking and debugging abilities.

In the advanced phase, the teaching will systematically guide students in AI-assisted programming training, including strategies for writing prompts, interpreting model responses, and optimizing code. Teachers will organize students to analyze and refine AI-generated code, gradually developing their ability to critically engage with AI. Finally, students will engage in human-machine collaborative development practices in open-ended tasks, completing a full programming process from requirement analysis, prompt writing, functionality implementation, to process documentation and reflection, enhancing their engineering application skills.

To support the implementation of the curriculum, this proposal adopts a multi-faceted teaching approach, including case-driven, group collaboration, and feedback loops. The teaching design integrates real-world

problems into classroom tasks, guiding students to build prompts and generate initial solutions. Students are encouraged to work in groups on human-machine co-creation tasks, fostering the integration of diverse perspectives through collaborative division of labor. A three-level feedback mechanism, consisting of AI model feedback, peer evaluations, and teacher feedback, will be established to enhance the interactivity and targeting of learning. Additional resources, such as prompt template libraries, error case collections, and AI debugging manuals, will be developed to support students in understanding AI model behavior and optimizing generation logic.

3.3. Evaluation mechanism and ability feedback

To comprehensively reflect students' ability development in an AI-collaborative environment, this proposal establishes a teaching evaluation system that combines "formative + summative + reflective" assessments, emphasizing the feedback loop between ability evaluation and thinking reflection. During the course, formative assessments will focus on the accuracy of prompt design, the quality of AI interaction, and the optimization process, providing teachers with data to guide the teaching process. Summative evaluation will be carried out through small projects or group tasks, assessing students' engineering abilities in functional implementation, structural design, and adherence to standards. Additionally, a reflective evaluation section will be included, where students write reports on their use of AI, analyzing the strengths and challenges of collaborating with AI and identifying possible improvement paths.

This multidimensional evaluation mechanism focuses not only on "what students did right" but also on "how they did it" and "how they can improve," building a feedback system that is oriented towards students' ability growth. It also helps teachers dynamically adjust teaching strategies, fostering a positive interaction between teaching and learning.

4. Feasibility and expected outcomes of the teaching reform

4.1. Teaching process and theoretical foundations

This teaching reform plan adopts a 15-week cycle, integrating the assistive capabilities of large language models into a progressive instructional process consisting of four stages: basic training, modular design, AI collaboration, and engineering practice.

In the first five weeks, instruction focuses on the fundamental syntax of C language, guiding students through concrete tasks to conduct preliminary prompt training and gradually build foundational skills in language expression and programming logic. Weeks six to ten emphasize structured programming and debugging, encouraging students to leverage AI for syntax and logic error detection, as well as for refining module design strategies.

Weeks eleven to fourteen are devoted to comprehensive project-based practice, during which students complete initial drafts with AI support, independently iterate and optimize their code, and document prompt design and collaborative workflows—thereby enhancing their abilities in system development and human–AI collaboration. In the final week, students present their projects and submit reflective reports on their use of AI, systematically reviewing their learning journey.

Visual Studio is recommended as the primary programming environment, complemented by web-based access to open-source LLMs such as DeepSeek. These tools support functionalities including auto-completion, debugging suggestions, and semantic code generation.

The instructional design is grounded in constructivist learning theory, emphasizing students' active

engagement and meaningful knowledge construction within task-driven contexts. Furthermore, insights from domestic and international studies on AI-assisted programming indicate that appropriately integrated AI tools can reduce beginner anxiety and entry barriers, while providing immediate feedback to enhance learners' diagnostic and reflective capabilities during debugging. Thus, from the perspectives of pedagogical structure, technical support, and cognitive development, the proposed teaching model demonstrates strong feasibility and theoretical soundness.

4.2. Evaluation of teaching outcomes and implementation challenges

Through the coordinated design of instructional goals and teaching processes, the reform initiative is expected to significantly enhance learning outcomes. On one hand, with AI assistance, students can improve both programming efficiency and code quality, leading to a more systematic understanding of syntax rules and structured expression. On the other hand, the process of prompt writing and iterative optimization fosters stronger problem awareness and continuous improvement capabilities. The reform also aims to stimulate students' intrinsic motivation, encouraging a shift from passive imitation to active design, and gradually cultivating essential skills for human–AI collaboration.

However, several implementation challenges may arise. First, the wide variation in students' programming abilities could pose difficulties; if the threshold for AI tool use is too high, less proficient students may become further marginalized. To address this, a tiered prompt task system can be introduced, along with personalized support resources, to lower the learning curve and ensure educational equity.

Second, LLMs can produce unstable or ambiguous outputs. If students overly rely on AI-generated content, their independent thinking abilities may be weakened. Therefore, the course should incorporate a "credibility alert" mechanism to guide students in critically evaluating AI responses.

Third, excessive dependence on AI might lead to a loss of control over programming details. To mitigate this, a prompt evaluation system centered on "refactoring and optimization" is recommended, emphasizing that AI outputs are references only, and requiring students to rewrite and reflect upon generated code.

In summary, this reform plan for High-Level Language Programming demonstrates high feasibility and innovation in terms of instructional design, theoretical underpinning, and technical deployment. It is expected to significantly improve the overall effectiveness of C language teaching. Nonetheless, careful attention must be paid to differentiated instruction and process management to avoid the risk of "outsourcing education to AI," thereby achieving a deep integration of AI-powered teaching, learning, and assessment.

5. Conclusion

The rise of large language models has brought new opportunities for the development of the High-Level Language Programming (C Language) course. Based on the concept of human–AI collaboration, this paper proposes a comprehensive teaching reform plan that integrates AI tools into the instructional process. The reform systematically addresses course objectives, content structure, teaching methods, and evaluation systems, aiming to enhance students' abilities in problem modeling, prompt design, and collaborative programming.

The core of this reform lies in combining traditional syntax training with intelligent assistance, fostering the integration of knowledge acquisition and capability development. Looking ahead, continued refinement of this approach may be achieved through pilot teaching programs, platform development, and interdisciplinary course integration. Such efforts will further advance the curriculum toward greater intelligence and convergence in the era of AI.

Funding

Education and Teaching Research Project of Beijing University of Technology (ER2024KCB08)

Disclosure statement

The authors declare no conflict of interest.

References

[1] Denny P, Kumar V, Giacaman N, 2023, Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language, Proceedings of the 54th ACM Technical Symposium on Computer Science Education, Vol 1, 1136–1142.

[2] Achiam J, Adler S, Agarwal S, et al., 2023, GPT-4 Technical Report. arXiv. https://doi.org/10.48550/arXiv.2303.08774

[3] Raihan N, Siddiq ML, Santos JCS, et al., 2025, Large Language Models in Computer Science Education: A Systematic Literature Review, Proceedings of the 56th ACM Technical Symposium on Computer Science Education, Vol 1, 938–944.

[4] Zhao Z, Tu Z, Sheng Y, 2024, Research and Practice on Teaching Reform of Basic Programming Course Empowered by Large Models. Progress in Modern Education, 2(14): 89–91.

[5] Lin J, Ni T, 2024, Exploration of Teaching Practice of C Programming Course under the Perspective of "Integration of Specialization and Innovation." Innovation and Entrepreneurship Theory Research and Practice, 7(16): 11.

[6] Wang Y, Mao G, Liao F, 2024, Exploration of "Problem-Driven" Teaching Mode of C Programming Course. Computer Education, 2024(8): 46–49.

[7] Zhu P, Hu X, Peng T, et al., 2024, Exploration of C Programming Teaching Practice for System Capability Cultivation. Software Guide, 23(8): 82–87.

[8] Su J, 2023, Research on Teaching of C Programming Course Integrated with Course Ideological and Political Education. Information and Computer (Theory Edition), 35(18): 244–247.