

Research on Applying Multimodal Large Language Models to Assist C++ Instruction

Ying Zhang, Yujun Zhang, Jiansheng Wu *

University of Science and Technology Liaoning, Anshan Liaoning, 114051, China

* Corresponding author: Jiansheng Wu (Email: ssewu@163.com)

Abstract: This research addresses prevalent issues in traditional C++ programming instruction, such as monotonous learning methods, the disconnect between theory and practice, and the lack of a robust assessment framework. It proposes an innovative AI-assisted teaching system that integrates multimodal large language models (MLLMs) with digital human technology. The system establishes an intelligent multimodal interaction framework, a dynamic knowledge graph, an adaptive learning engine, a real-time code interaction module, and a practical task and progress tracking mechanism. This integrated approach enables comprehensive coverage of programming concepts, personalized instruction, and real-time code diagnostics, ultimately enhancing students' practical programming skills. A survey-based methodology was employed to evaluate the system's projected effectiveness. The results indicate that the system holds significant potential for improving learning efficiency, enriching the learning experience, and facilitating deeper knowledge acquisition. Consequently, it presents a promising solution for innovating C++ programming pedagogy.

Keywords: Intelligent Teaching; Multimodal Large Language Models (MLLMs); C++ Programming Education.

1. Introduction

C++ programming education serves as a fundamental cornerstone within the field of computer science, playing a critical role in cultivating students' logical programming thinking, code construction capabilities, and engineering practice literacy. Against the backdrop of accelerating digital transformation, the limitations of traditional C++ teaching methods have become increasingly apparent[6], failing to meet the urgent societal demand for high-quality programming talent. From the perspective of modern educational approaches such as maker education, the reform of programming instruction emphasizes the critical importance of cultivating practical and innovative capabilities[1]. The advancement of Large Language Models (LLMs) and Artificial Intelligence (AI) technologies presents a novel possibility for the evolution of programming education. An intelligent AI-assisted C++ teaching system, built upon multimodal large language models and digital human technology, holds the potential to break through the bottlenecks of traditional pedagogy and inject new vitality into programming instruction.

Globally, research on the application of AI in programming education has yielded significant results, leading to the emergence of several representative intelligent tutoring platforms. For instance, platforms like Codecademy utilize AI algorithms to provide intelligent code analysis, capable of identifying errors in student code in real-time and offering targeted repair suggestions, as well as generating personalized learning paths based on student progress and performance. DataCamp, an institution focused on data science programming education, employs AI technology to implement features such as intelligent code completion and personalized learning content recommendations, enabling students to quickly acquire skills in data processing and analysis. While these platforms have achieved a certain degree of success in programming education, they predominantly focus on more general-purpose languages like

Python. Their support for teaching complex concepts inherent to C++, such as memory management, pointer manipulation, and object-oriented programming, remains relatively weak, insufficient to address the in-depth requirements of C++ education.

Domestically, the reform and innovation of C++ programming education are also gradually advancing. Some universities and educational institutions have begun integrating AI technologies into C++ instruction, establishing intelligent code evaluation systems and building online programming learning platforms. However, current related research in China is still in an exploratory phase[4]. Most projects remain at the preliminary stage of technology application and have not yet evolved into systematic pedagogical frameworks or innovative practical paradigms. Existing AI-assisted teaching systems exhibit shortcomings in areas such as explaining C++ programming concepts, diagnosing code errors, and planning learning pathways, failing to provide students with personalized and in-depth instructional support.

The objective of this study is to develop an intelligent AI-assisted C++ teaching system that integrates multimodal large language models with digital human technology. Generative artificial intelligence serves as the central driving force in building new models of programming pedagogy[2]. It aims to break through the constraints of limited interactivity and insufficient practical components in traditional programming education, thereby creating an intelligent and personalized programming learning environment. This research will proceed from the following aspects: firstly, constructing a comprehensive multimodal intelligent interaction framework that supports students in interacting with a digital tutor through various means such as voice queries[5], code screenshot uploads, and text input, thereby facilitating immediate communication and problem-solving during the learning process. Secondly, developing an interactive module with real-time code diagnosis and repair capabilities. This module will utilize Large Language Models (LLMs) and code

analysis techniques to perform real-time analysis of student-written C++ code, identifying syntax errors, logical flaws, and performance issues, and providing detailed error analysis alongside step-by-step repair suggestions. This will enable students to quickly pinpoint and resolve problems. Thirdly, establishing a practical task and learning progress tracking system. This involves designing a sequence of C++ programming tasks of progressive difficulty, dynamically adjusting task complexity based on individual student performance, continuously recording learning progress and outcomes, and providing comprehensive data support for instructors. This data-driven approach empowers educators to implement targeted pedagogical improvements.

The creation of an intelligent AI-assisted C++ teaching system leveraging multimodal large language models and digital human technology can introduce transformative changes to C++ programming education. This system not only holds the potential to enhance the student learning experience and programming competency but also provides instructors with an efficient and precise teaching tool. It represents a step towards more intelligent and personalized C++ education, creating favorable conditions for cultivating the high-caliber programming talent required in the digital era.

2. C++ Teaching System Based on Multimodal Large Language Models

2.1. Overall System Architecture Design

The system adopts a layered architecture to ensure well-defined functionalities and efficient collaboration among its modules. This architecture consists of three primary layers: the Data Layer, the Service Layer, and the Application Layer, as illustrated in Figure 1. The Data Layer manages the storage of information, including the C++ programming knowledge base, student learning behavior records, and system configurations. The Service Layer provides core services, such as multimodal large language model services, digital human interaction services, code analysis tools, and adaptive learning functionalities. The Application Layer offers a user-friendly access point and operational interface for both students and instructors via a web-based platform, serving as the central hub for user interaction.

The advantage of this layered architecture lies in its low coupling between layers, which significantly enhances flexibility during system expansion and maintenance, while reducing the complexities of interdependencies. The Data Layer employs a distributed database architecture, storing the C++ programming knowledge base, student learning behavior data, and system configuration information in separate databases to improve system security and data access efficiency. The Service Layer utilizes a microservices architecture, decomposing the system's core functionalities into independent microservices, each focused on a specific function. This approach facilitates independent development, deployment, and maintenance. The Application Layer adopts a responsive design, ensuring a seamless user experience across both desktop and mobile devices. This design not only optimizes efficiency but also enables personalized, device-specific interactive experiences.

2.2. Multimodal Intelligent Interaction System

The system allows students to interact with the digital tutor through multiple channels, including voice queries, uploaded code screenshots, and text input. Leveraging advanced speech

recognition, Natural Language Processing (NLP), and image recognition technologies, the digital tutor accurately understands the intent behind student inquiries and responds using various modalities, such as voice, text, graphics, and code examples. For instance, when a student asks about implementing polymorphism in C++ via voice, the digital tutor not only provides a clear explanation of the basic concept and implementation methods but also supplements the response with relevant code examples to help students better grasp the topic.

This multimodal intelligent interaction framework moves beyond the single-mode interactions typical of traditional instruction[5], aiming to deliver a more natural and convenient learning experience. The speech recognition technology uses state-of-the-art deep learning models that accurately recognize speech inputs with different accents and speaking rates, ensuring reliable comprehension despite variations in pronunciation. NLP technology, through semantic understanding and matching with the knowledge graph, interprets the intent behind student questions and provides precise answers. Meanwhile, image recognition technology extracts and parses code content from uploaded screenshots, identifies potential issues, and offers constructive feedback, thereby enhancing learning efficiency.

2.3. Dynamic Knowledge Graph and Personalized Learning Engine

The system organizes the core knowledge framework of C++ programming into a structured and dynamically updated knowledge graph[3]. This graph not only covers fundamental concepts, syntax rules, algorithm implementations, and practical applications of the C++ language but also connects different knowledge points through semantic relationships. Based on students' learning data—including answer accuracy, time spent on learning activities, and interaction records—the system personalizes and adjusts each student's learning path. The adaptive learning engine dynamically modifies the learning plan according to individual needs. When the system detects that a student has a weaker grasp of a particular concept, it automatically recommends related instructional videos, exercises, and case studies, gradually increasing the learning difficulty to help the student solidify their foundation and ultimately enhance their overall competency.

The dynamic knowledge graph is constructed using ontology and semantic web technologies, abstracting C++ programming knowledge into entities, attributes, and relationships. These elements are then linked through semantic associations to form a comprehensive knowledge network. The knowledge graph's updating mechanism combines automated crawling with manual review to ensure accuracy and timeliness. The personalized learning engine uses machine learning algorithms to analyze students' learning behavior data, build individualized learning models, and generate customized learning pathways tailored to their specific needs, thereby supporting students in efficiently mastering programming skills.

2.4. Real-time Code Interaction and Diagnostics Module

The system's built-in real-time code interaction module dynamically analyzes student-written code, accurately identifying syntax errors, logical issues, and potential performance bottlenecks. When errors are detected, the digital tutor provides a detailed analysis of the root cause,

along with step-by-step repair suggestions. For example, if a student's code contains an infinite loop, the digital tutor not only pinpoints the exact location of the issue but also guides the student through the process of revising the code, helping them resolve the problem incrementally. Additionally, the digital tutor conducts in-depth performance analysis and offers optimization recommendations, enabling students to write more efficient and maintainable code, thereby enhancing both their programming skills and quality awareness.

The real-time code interaction module combines static code analysis with dynamic execution. Using syntax parsing and semantic analysis, it detects errors before execution, preventing their accumulation. Dynamic execution identifies runtime issues such as performance bottlenecks and logical flaws. When providing repairs, the digital tutor guides students stepwise, helping them understand error causes and solutions. This strengthens both problem-solving abilities and programming techniques.

2.5. Practical Task and Learning Progress Tracking Mechanism

The system designs programming tasks that progress from foundational to advanced levels, allowing students to continually enhance their programming skills through structured, incremental learning. These tasks begin with basic syntax exercises and gradually guide students toward mastering more complex concepts and techniques. Task difficulty is dynamically adjusted based on performance metrics such as answer accuracy and completion status, ensuring that the challenges remain appropriate without being overwhelming or discouraging. Upon completing each task, the system records the student's learning progress and generates visual reports, including knowledge mastery curve charts and learning progress line charts, to clearly illustrate learning outcomes and development over time.

Teachers can monitor student progress in real time through a data visualization interface, allowing for timely, personalized interventions. Those struggling receive targeted support, while advanced learners are challenged with more complex tasks. This ensures all students develop programming skills at an appropriate pace, enhancing overall learning outcomes.

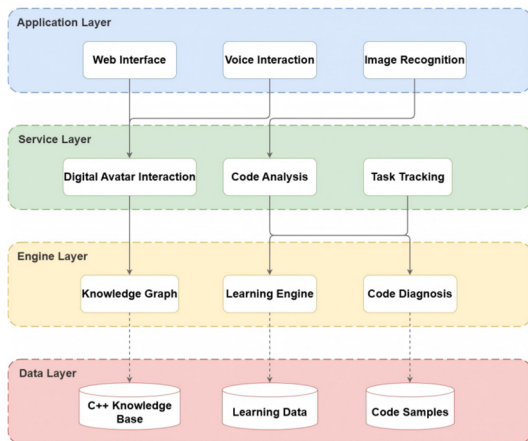


Figure 1. Overall System Architecture

3. Results and Analysis

3.1. System Functionality Implementation

Through this research, we have successfully developed an

intelligent C++ teaching assistance system that integrates multimodal large language models with digital human technology. The system not only incorporates core functions, such as multimodal intelligent interaction, a dynamic knowledge graph paired with an adaptive learning engine, and real-time code interaction, but also enhances students' programming learning outcomes through practical task systems and progress tracking mechanisms.

By supporting flexible interaction methods, the system enables students to engage with the digital tutor in multiple formats—including voice, text, and images—during the learning process. This facilitates the resolution of challenging programming issues and provides personalized learning pathways. The dynamic knowledge graph automatically adjusts the learning content based on each student's progress and performance, ensuring that they acquire programming skills at a pace suited to their individual learning rhythm.

Simultaneously, the real-time code interaction module instantly recognizes and analyzes student-written code, offering detailed diagnostics and repair suggestions. This enhances students' programming competence and problem-solving abilities. Through these innovative features, the system not only provides comprehensive support for programming learning but also significantly elevates the level of instructional intelligence. It meets the needs of learners at various stages and with diverse demands, further advancing the personalization and precision of programming education.

3.2. Application Effectiveness Evaluation

To conduct a comprehensive evaluation of the C++ intelligent assisted teaching system, this study selected undergraduate students from the School of Computer Science and Software Engineering at the University of Science and Technology Liaoning as experimental subjects. Employing a controlled experimental method, students were randomly assigned to either an experimental group or a control group. Students in the experimental group fully utilized the system's various functions—including multimodal interaction, intelligent diagnostics, and personalized learning path planning—throughout their C++ programming coursework. Students in the control group followed traditional teaching methods, consisting primarily of instructor-led classroom lectures and post-class assignment exercises.

Based on a comprehensive evaluation of the two student groups, a detailed analysis was conducted from three aspects: academic performance, practical programming ability, and learning interest. The results revealed that students in the experimental group achieved a significantly higher overall average score in the final examination compared to the control group. Specifically, the average score for the experimental group was 85.3 points, while the control group averaged 75.6 points, representing a 12.7% advantage for the experimental group. Regarding programming projects, students in the experimental group showed stronger performance, completing 78.5% of projects, 21.3 percentage points higher than the control group's 57.2%. Furthermore, students in the experimental group showed significant improvement in practical programming skills: 84% of them were capable of independently completing relatively complex programming tasks, compared to only 60% in the control group.

In terms of learning attitude, the experimental group displayed greater proactivity and initiative. Survey data indicated that 93% of students in the experimental group

expressed willingness to continue learning programming outside of class hours, considerably higher than the 76% in the control group. Students in the experimental group generally exhibited stronger self-directed learning ability and independent thinking, with over 80% reporting that they could independently resolve issues encountered during programming, compared to 62% in the control group.

According to the questionnaire survey, students in the experimental group demonstrated significantly higher interest in learning C++ programming. Specifically, 82% of experimental group students expressed strong interest in C++ programming and a desire to pursue further in-depth study of the course, while the corresponding figure in the control group was only 58%. Additionally, 65% of students in the experimental group indicated a willingness to actively explore programming knowledge and techniques, including more advanced programming concepts, far surpassing the 45% in the control group. During the learning process, students in the experimental group also showed markedly higher levels of engagement and interaction: 92% reported frequently participating in classroom interactions and asking questions, compared to 68% in the control group.

These quantitative data underscore that students in the experimental group achieved significant improvements across multiple dimensions—including academic performance [4], practical programming ability, and learning interest—further validating the notable advantages of the intelligent AI-assisted C++ teaching system proposed in this study in enhancing instructional effectiveness and stimulating student motivation.

3.3. Analysis of System Advantages

Compared to traditional C++ programming education, this system offers the following distinct advantages:

(1) Personalized Learning: The system tailors individual learning paths and resource recommendations for each student based on their learning behaviors and needs. Such personalized learning plans can effectively enhance learning efficiency and assist students in overcoming encountered difficulties.

(2) Real-time Feedback: The system possesses robust real-time code analysis capabilities, enabling it to instantly parse student-written code and accurately identify syntax errors, logical flaws, and performance bottlenecks. Following detailed error diagnosis and repair suggestions, students can quickly locate and correct their mistakes, preventing the accumulation of errors and thereby improving their programming skills.

(3) Multimodal Interaction: To provide learners with a more natural and convenient learning experience, the system supports various interaction modes, including voice, image, and text. Students can obtain feedback via voice queries or text input and can also interact with the digital avatar by uploading code screenshots. This allows them to choose the most suitable self-learning method based on their personal habits[5].

(4) Data-Driven Insights: The system continuously collects student learning data, such as answer accuracy rates, time spent on topics, and interaction records. Leveraging this data, instructors can obtain more detailed academic analysis reports. The insights derived from this data provide instructors with precise strategies for pedagogical intervention, thereby improving individual student learning outcomes and

enhancing overall teaching effectiveness[7].

By implementing these innovative functionalities, this system addresses many of the challenges inherent in traditional C++ programming instruction. It also provides robust support and a novel approach for personalized and intelligent programming education.

4. Conclusion

The C++ teaching assistance system integrates multimodal large language models with digital human technology. It establishes a multimodal intelligent interaction framework, a dynamic knowledge graph with an adaptive learning engine, a real-time code interaction module, and a system for tracking practical tasks and learning progress. This comprehensive approach covers all facets of programming learning, providing learners with a personalized and precise learning experience while promoting real-time code diagnostics and the enhancement of practical skills. The research findings demonstrate that this system can significantly improve the effectiveness of C++ programming instruction and enrich the learning experience, offering a novel pathway toward the intelligentization of programming education.

Acknowledgments

The authors sincerely acknowledge the financial support provided by the USTL College Student Innovation and Entrepreneurship Training Program project titled "An Intelligent C++ Teaching Assistance System Based on Multimodal Large Language Models".

We also gratefully acknowledge the support from the 2025 Liaoning Provincial Undergraduate Teaching Reform Research Project titled "Industry-Education Collaboration · Virtual-Real Integration · Personalized Development: Research and Practice on the Training Model of Applied Computer Talents in Local Universities".

References

- [1] NAN Nan, NAN Chao. Study and application of computer programming from the perspective of maker education in colleges and universities [J]. *Wireless Internet Technology*, 2021, 18(15): 130-131.
- [2] TANG Q W. Research on the construction of high school programming teaching mode supported by generative artificial intelligence [D]. Yangzhou University, 2025.
- [3] ZHAO Mankun, ZHAO Yue, LIU Zhiqiang, XU Tianyi, WANG Jianrong, YU Mei, YU Ruiguo. Cheng Tian Xue Platform: Innovative practice of teaching reform in programming courses in the intelligent era [J]. *Computer Education*, 2025, (08): 16-21.
- [4] ZHOU Yingchun. Intelligent reconstruction of primary school programming curriculum driven by generative AI: example discussion on innovative practice of LLM assisted C++ school-based curriculum [J]. *Innovative Talent Education*, 2025, (04): 51-55.
- [5] CHEN Hongcai. Application of virtual digital human technology in the information interaction of online education platform [J]. *Wireless Internet Technology*, 2024, 21(24): 68-71.
- [6] Lokare T V, Jadhav M P, Patil S S. An integrated approach for teaching object oriented programming (C++) course [J]. *Journal of Engineering Education Transformations*, 2018, 31(3): 17-23.
- [7] Serhani M A, Bouktif S, Al-Qirim N, El Kassabi H T. Automated system for evaluating higher education programs [J]. *Education and Information Technologies*, 2019.