Journal of Contemporary Educational Research, 2026, Volume 10, Issue 2
http://ojs.bbwpublisher.com/index.php/JCER
ISSN Online: 2208-8474
ISSN Print: 2208-8466

Exploration of Blended Teaching Practice in the Course of Fundamentals of Programming

Yongrui Cui

School of Computer Science and Engineering, Dalian Minzu University, Dalian, Liaoning, China

Abstract: The "Fundamentals of Programming" course employs a blended learning model that integrates online resources with offline instruction to enhance students' self-directed learning and practical skills. Powered by knowledge graphs, the curriculum enables personalized learning paths where students autonomously design their study plans, while instructors dynamically adjust teaching strategies based on learning data to achieve precision education. The gamified practice platform transforms grammar training into immersive gaming experiences, significantly boosting learning engagement and practical retention. The teaching strategy adopts tiered cultivation, combining macro-level projects with micro-level knowledge points to strengthen computational thinking and practical abilities. Implementation results demonstrate improved mastery of knowledge points, continuous enhancement of coding and debugging skills, and markedly strengthened learning motivation. This model creates a closed-loop system for knowledge transfer and competency development, laying a solid foundation for subsequent specialized courses. It embodies innovative concepts of knowledge graph-driven learning, gamified practice, and tiered cultivation, effectively fostering students' self-directed learning drive and critical thinking.

Keywords: Blended learning; Fundamentals of Programming; Knowledge graph; Gamified learning; Computational thinking

Online publication: March 12, 2026

1. Introduction

As a core course in software engineering, "Fundamentals of Programming" plays a pivotal role in cultivating computational thinking and programming fundamentals. Traditional teaching methods, however, often emphasize one-way knowledge transmission while neglecting self-directed learning and practical skill development, leaving students ill-equipped to apply knowledge to real-world problem-solving. The blended learning model, through the seamless integration of online resources and offline instruction, offers innovative approaches to curriculum reform. This paper explores the innovative implementation of this model in the course to enhance teaching quality and nurture technical professionals with practical innovation capabilities and self-directed learning awareness.

The blended learning model has achieved remarkable progress in the foundational programming course. In

the "SPOC + online resources" hybrid approach, SPOC-based blended instruction effectively enhances students' programming skills in courses like "C Programming" through the synergy of online resources and offline classes [1]. Meanwhile, the "MOOC + SPOC" model has demonstrated its effectiveness in boosting engagement in courses such as digital circuits, though challenges in online-offline coordination and learning continuity remain to be addressed [2]. In the "Smart Monitoring + Problem-Based Guidance" innovation model, intelligent question banks enable learning process monitoring (e.g., in operating system courses), yet it addresses the issue of insufficient self-directed learning motivation among students. The problem-guided blended teaching approach cultivates students' problem-solving abilities, but question design must align with students' cognitive levels [3–5]. In the "Output-Oriented + Resource Integration" deepening model, the "output-oriented method" emphasizes learning outcomes, requiring exploration of scientific evaluation methods. Online-offline resource integration should avoid redundant construction and establish a resource evaluation system. The Python experiment model based on micro-lectures achieves precise reinforcement through structured chapter systems, but improves in micro-lecture production efficiency [6,7]. In the "Technology Empowerment + Environment Optimization" support model, the live-streaming blended model needs to address the impact of the network environment on teaching experience. The "Large Model-Assisted" dual-engine model requires balancing cost and efficiency [8–10].

2. Development of online teaching resources

The cornerstone of blended learning is developing online resources, as demonstrated in the following examples.

2.1. Video resources

The video resources consist of two components: micro-lectures and classic exercise explanations. A total of 61 videos spanning 1,619 minutes (26 hours) are distributed across 11 chapters and 49 sections of teaching materials, with designated task points for students to independently watch and study.

2.2. Courseware and document resources

Featuring 15 premium teaching materials and 20 chapter lesson plans spanning over 500 pages, this resource pack delivers rich, visually engaging content that aligns seamlessly with the curriculum. Distributed across online chapters, it includes task-based learning points designed for self-directed study.

2.3. Question bank resources

A self-developed collection of 240 test questions, featuring diverse formats and rich content, distributed across online chapters with designated task points for students to practice and assess independently.

2.4. Knowledge graph

A self-developed comprehensive knowledge graph for courses, containing 144 knowledge points with a resource relevance rate of 90.28%, designed for students' self-directed learning.

Students actively utilized online resources for self-directed learning, demonstrating high engagement. In the most recent teaching cycle alone, 144 participants enrolled in the course, with 46,257 classroom activities and 223,790 classroom visits recorded. On average, each participant participated in 321 activities and accessed the classroom 1,554 times.

3. Blended learning design

3.1. Course positioning and student analysis

This course serves as a foundational core course for software engineering majors, designed to cultivate computational thinking and establish programming fundamentals. Using C as the primary language, it covers essential programming concepts, principles, and methodologies, including core topics such as basic data types, control structures (sequential, branching, and looping), arrays, functions, structures, and file handling.

This course is designed for first-year software engineering students who possess basic IT literacy but lack computational thinking and systematic programming training. They are typically self-motivated learners with strong self-study capabilities, highly receptive to gamified and interactive teaching methods, and exhibit significant individual differences with varying foundational levels. Traditional classroom instruction cannot effectively accommodate the diverse learning needs of students at different proficiency levels.

3.2. Overall framework of blended learning

The blended learning framework of this course is structured as a "three-stages" + "dual-tracks" model. The "three stages" comprise three progressive phases: pre-class online self-study, classroom-based blended instruction, and post-class extended learning. The "dual tracks" involve the theoretical component from Chaoxing Smart Course and the practical component from Toge Practice Platform, operating in parallel throughout the instructional process. Online instruction emphasizes knowledge delivery and foundational cognitive development, while offline sessions focus on skill cultivation and critical thinking enhancement. The practice platform provides real-time feedback and automated assessments, effectively supporting the "learn-practice-test-evaluate" closed-loop cycle.

3.3. Design of online teaching system

3.3.1. Theoretical course development on Chaoxing Platform

Leveraging the Chaoxing Smart Teaching Platform, we have developed an integrated online theoretical course system that combines micro-lecture videos, structured chapters, intelligent quizzes, and knowledge graph applications.

We have built a micro-lecture repository featuring 144 knowledge points, each lecture lasting 5–12 minutes. The structure follows a "problem introduction + principle explanation + case demonstration + summary and enhancement" format, catering to students' fragmented learning habits. The platform supports playback at different speeds and allows resuming from any point.

The course is structured into 10 instructional units and 1 exercise unit, systematically covering fundamental concepts including: software engineering principles, computer system architecture, programming fundamentals, algorithm design basics, C language fundamentals, basic control structures, data type construction (arrays, structures, unions, enumerations), functions, compiler preprocessing and custom types, pointers, and data files. Each chapter features clear learning objectives, knowledge map navigation, key concept highlights, and intelligent unit tests, with well-defined task points to create a complete learning cycle.

Develop an intelligent question bank and assignment system. The system features 240 questions with tiered difficulty levels, including multiple-choice, true/false, fill-in-the-blank, and programming exercises, covering all key knowledge areas. It supports automated test generation and push notifications, while assignments receive AI-powered grading and personalized feedback, enabling personalized, automated, and precise formative assessment.

Developing knowledge graph applications. A knowledge graph containing 144 key concepts was constructed based on the curriculum framework, achieving over 90% resource connectivity and comprehensive coverage of all knowledge points. Students can use the graph to track their mastery of prerequisite knowledge and identify dependencies on subsequent concepts. Teachers can monitor real-time class-wide comprehension

heatmaps and individual learning gaps, enabling dynamic adjustments to teaching strategies.

3.3.2. Development of the Touge Practice Platform

Building on the Touge Practice Platform, we have established an integrated online training system combining task-driven learning, real-time feedback, and skill advancement.

Developing a tiered project framework. Catering to students' gamified learning preferences, the course design incorporates 32 practical projects that comprehensively cover all curriculum knowledge points, featuring 50 progressively challenging levels. The difficulty progression is systematically structured from beginner-friendly to advanced, with each level providing detailed implementation guidelines, test case explanations, and an integrated online compilation system that supports code execution, debugging, and real-time error alerts. Students receive instant feedback within seconds after submitting their code. Each project includes a leaderboard and achievement system to foster intrinsic motivation for learning.

The intelligent evaluation system supports batch verification and smart assessment of multiple datasets, generating personalized evaluation reports for each student. It scores based on code quality and allows self-regulation of multiple evaluation criteria, including coding standards and grammar rules. This significantly enhances students' self-awareness and correction skills in coding while substantially reducing teachers' grading workload.

3.4. Design of offline teaching implementation

3.4.1. Macro-micro knowledge system layered task-driven instruction

The curriculum incorporates over 20 classic project cases, such as student grade management systems and contact lists, as macro-level project vehicles. These projects follow standard software development processes, covering critical stages including requirements analysis, system design, coding implementation, and testing. This approach comprehensively integrates the entire knowledge framework of the course. By breaking down micro-level knowledge points into small project cases, the teaching model adopts a dual-track approach: macro-level projects guiding the way while micro-level projects reinforce understanding. This methodology effectively enhances students' comprehension and mastery of software engineering principles and practical skills.

3.4.2. Classroom teaching reconstruction

As the core domain of blended learning, offline classrooms should not be isolated from online instruction but rather deeply integrated with digital resources to mutually reinforce each other. Teachers can leverage platform data to pinpoint students' common challenges before class, deliver targeted explanations during lessons to enhance efficiency, and increase collaborative training components. This approach facilitates the transition from knowledge transmission to competency internalization.

For an 80-minute lesson, the class will follow a four-step structure: assessment, instruction, practice, and evaluation.

Pre-class assessment (10 minutes): Administer a pre-class quiz via the Chaoxing Platform to quickly evaluate students' learning outcomes and adjust teaching priorities accordingly.

25-minute intensive lecture: Targeting key challenges from pre-class preparation and pre-test feedback, the session employs diverse teaching methods to thoroughly analyze and explain these concepts, drawing on both the instructor's expertise and the syllabus requirements.

In-class collaborative practice (30 minutes): Building on the course content, students utilize the Chaoxing Smart Course Platform to complete group tasks. They discuss solutions, code on the spot, and conduct peer reviews and checks, fostering collaborative learning.

Lesson wrap-up (15 minutes): The instructor provides real-time feedback and summarizes student collaboration outcomes based on platform data, identifies common challenges, and demonstrates optimization strategies. Using platform data, they select exemplary code for on-the-spot review and offer constructive encouragement.

4. Features and outcomes

4.1. Innovative features of the course

Knowledge graph-driven personalized learning: Through detailed modeling of 144 knowledge points, it enables intelligent learning path planning and targeted reinforcement of weak areas. The comprehensive and stratified learning data allows teachers to maintain a holistic understanding of their students while providing learners with clear insights into their own progress.

Deep integration of gamified learning platforms: By adopting the challenge-based system from platforms like Touge or Chaoxing Smart Class, the platform transforms tedious grammar drills into immersive, engaging games, significantly boosting students' interest and retention in learning.

Seamless integration of online and offline: Digital resources and physical courses are no longer merely additive or isolated, but rather achieve closed-loop teaching evolution through data-driven feedback and optimization.

Hierarchical cultivation of computational thinking through macro-micro integration: From macro-level understanding of software processes to micro-level mastery and practice of programming knowledge; From memorization of syntax rules to algorithm design and application, and further to complex system development, achieving the enhancement of computational thinking capabilities.

4.2. Teaching effectiveness

4.2.1. Quantitative assessment of learning outcomes

After adopting the blended teaching model, data tracking of the self-built knowledge graph covering 144 key concepts revealed an average student mastery rate of 88.36%. This represents an 8-percentage-point improvement from last year's 79.3% derived from classroom test data. Notably, abstract concepts like functions and pointers showed significant comprehension gains through repeated online self-study via micro-lecture videos, as illustrated in **Figures 1** and **2**.

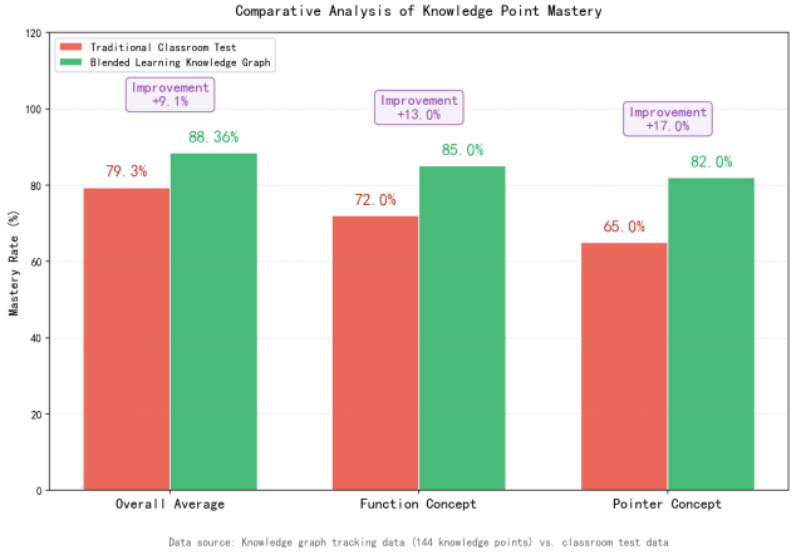

Figure 1. Comparative analysis of average mastery rates

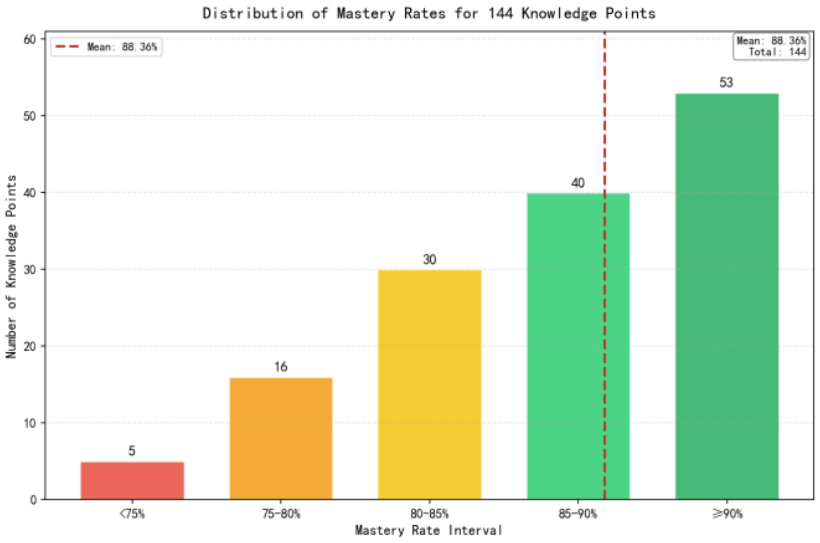

Figure 2. Distribution of mastery rates across 144 concepts

Data from the Touge Practice Platform indicates an overall pass rate of 97% across 32 projects and 50 levels (including 6% for catch-up levels). Performance evaluation results showed an excellence rate of 21% and a good rate of 70%, as depicted in **Figure 3**. As teaching progressed, the complexity of project codes increased, leading to higher average evaluation attempts per level. However, the pass rate showed no cliff-like decline, demonstrating students' adaptation to the game-based learning model and sustained motivation. Code debugging skills and quality continued to improve. Compared to traditional offline classroom teaching, students' average code output per person increased significantly. Most students could independently complete over 200 lines of code, with top performers showcasing strong collaboration and coding abilities during semester-end project presentations, earning widespread praise from both teachers and peers. Some levels presented extreme challenges, with students requiring 54 attempts to pass, prompting personalized guidance from instructors.

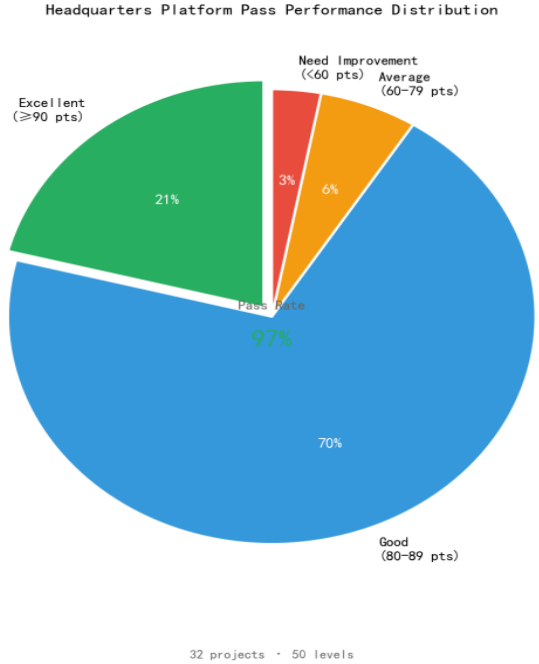

Figure 3. Distribution of Touge Platform performance metrics

4.2.2. Learning behaviors and competency enhancement

The learning behavior statistics from the Chaoxing Smart Course Platform reveal profound changes in student engagement following the implementation of blended learning. Chapter completion rates reached 97.29%, with homework averages hitting 90 points and chapter quizzes averaging 88 points. The rising proportion of students revisiting videos indicates significantly enhanced self-motivation in autonomous learning. The increased retake rate of chapter quizzes demonstrates students' willingness to attempt multiple attempts for higher scores, further reflecting strengthened intrinsic motivation. This elevated drive will undoubtedly lay a solid foundation for mastering core professional courses in subsequent stages.

The layered case-based teaching design integrating macro and micro perspectives, coupled with the effective implementation of online preview, enabled students to demonstrate strong communication and expression skills during group collaboration in offline classes. The code peer review and final project presentation sessions fostered a lively atmosphere while also providing ample opportunities for critical thinking development.

5. Conclusion

This study presents an innovative blended teaching model for the "Fundamentals of Programming" course that effectively integrates online and offline instructional resources. By leveraging knowledge graphs to enable personalized learning paths, implementing gamified practice platforms to enhance student engagement, and adopting a tiered cultivation approach combining macro-level projects with micro-level knowledge points, the model successfully addresses the limitations of traditional one-way knowledge transmission. The implementation results demonstrate significant improvements in knowledge mastery rates, high completion rates, and sustained learning motivation. This closed-loop teaching system not only strengthens students' computational thinking and practical programming abilities but also cultivates self-directed learning capabilities and critical thinking skills. The proposed model provides a replicable framework for programming education reform and lays a solid foundation for subsequent specialized courses in software engineering curricula. Future work will continue to optimize the intelligent teaching system and explore deeper integration of emerging technologies to further enhance teaching effectiveness and student learning experiences.

Funding

This study was supported by the General Project of Educational and Teaching Reform Research of the State Ethnic Affairs Commission of China in 2025 (No. 2025-GMJ-233): Integration of Intelligent Teaching and Education & Jointly Forging a Shared Future: Research on the Digital Teaching Reform and Collaborative Innovation of Multidimensional Resource Bank for the Course of An Introduction to the Chinese Nation Community, and supported by the 2025 Teaching Reform Project of Dalian Minzu University (No. YB202554).

Disclosure statement

The author declares no conflict of interest.

References

[1] Ning X, Zhang X, 2026, Research and Practice on Blended Teaching Model for Programming Courses Based on SPOC-A Case Study of "C Language Programming" Course. Industry and Information Technology Education, (02): 52–55 + 71.

[2] Shi C, Jiao J, 2025, Application of MOOC+SPOC Blended Teaching Model in Digital Circuit Course. Journal of Changchun Normal University, 44(06): 147–149.

[3] Chen Y, Ma L, 2023, Exploring and Practicing Blended Teaching of Operating Systems Based on "MOOC+SPOC." Science, Education and Culture Digest, (17): 103–106.

[4] Li J, 2023, MOOC+SPOC Teaching Practice in Computer Courses. Electronic Technology, 52(07): 319–321.

[5] Hu Y, Geng Z, Pu Y, et al., 2022, Exploring a Problem-Driven Blended Online-Offline Teaching Model. Computer Education, (02): 73–78.

[6] Li X, 2021, Research and Practice on the SPOC Hybrid Teaching Model of College English Based on the "Output-Oriented Approach" in the Context of Educational Informatization. Foreign Trade and Economics, (11): 144–147.

[7] Cui M, Liu Y, Zhao B, et al., 2025, Application of Blended Teaching in Clinical Teaching of Orthopedic Surgery: A Case Study of Yutang + DingTalk Live. Education and Teaching Forum, (15): 169–172.

[8] Li J, 2026, Reform and Practice of C Language Teaching Mode Driven by Dual Engines: Online-Offline Hybrid + Large Model-Assisted. Software Guide, 1–19.

[9] Shen Y, 2023, Teaching Optimization of "C Programming" Course from the Perspective of Blended Learning. Information System Engineering, (05): 160–162.

[10] Qi J, Sun K, 2025, Design and Practice of a Blended Teaching Model for Python Programming Experiments Based on Micro-Lectures. China Educational Technology Equipment, (24): 129–132.