



Research on the Teaching Reform and Practice of C Language Course Based on Project-driven

Jun Xu

School of Information Technology and Engineering, Guangzhou College of Commerce, Guangzhou 511363, Guangdong, China.

How to cite this paper: Jun Xu. (2025) Research on the Teaching Reform and Practice of C Language Course Based on Project-driven. *Advances in Computer and Communication*, 6(5), 317-320.
DOI: 10.26855/acc.2025.12.010

Received: November 5, 2025
Accepted: December 2, 2025
Published: December 30, 2025

***Corresponding author:** Jun Xu, School of Information Technology and Engineering, Guangzhou College of Commerce, Guangzhou 511363, Guangdong, China.

Abstract

This study investigates the pedagogical reform of C language instruction, addressing challenges in traditional teaching such as monotonous methodologies, disconnection between theory and practice, and flawed evaluation systems. Through a randomized design, two second-year Computer Science and Technology classes were selected as experimental units, with the experimental group adopting project-based learning while the control group maintained conventional teaching approaches. The implementation process encompassed project selection, knowledge delivery, project execution, feedback assessment, and evaluation. Results demonstrated that the experimental group achieved significantly higher academic performance and practical competencies compared to the control group, while also exhibiting greater engagement in learning attitudes, including interest, initiative, teamwork willingness, and course satisfaction. These findings indicate that project-based learning effectively enhances C language instruction outcomes, providing valuable insights for curriculum reform.

Keywords

Project-driven; C language course; teaching reform; practical case

1. Introduction

As a vital programming language, C plays a pivotal role in computer science curricula. It not only serves as the foundation for students to master coding skills but also acts as a key pathway to develop logical thinking and problem-solving abilities. However, traditional C language teaching methods have revealed numerous shortcomings in practice, failing to meet modern education's demands for cultivating students' comprehensive competencies.

2. Problems in the Traditional C Language Classroom Teaching

2.1 The Teaching Methods are too Limited

In traditional C language instruction, teaching methods remain largely monotonous, predominantly relying on one-way lecturing by instructors. Teachers deliver detailed explanations of grammar rules and sentence structures in textbook chapter order from the podium, leaving students in a passive learning state. This "cramming" approach creates a dull and oppressive classroom atmosphere, lacking interaction and vitality [1].

2.2 Theoretical and Practical Disconnection

In theoretical instruction, teachers prioritize systematic and comprehensive knowledge delivery, dedicating substantial time to explaining fundamental concepts like C language syntax, data types, and control structures. While these fundamentals are essential for students to build programming frameworks, the excessive focus on theoretical

derivations and written explanations often leads to rote memorization of knowledge points without a genuine understanding of their practical applications in real-world programming.

3. Practice Object

3.1 Experimental Class

The experimental class comprises 45 students, including 30 males and 15 females. With an average score of 75 points from their foundational computer courses, the students demonstrate balanced performance across all score ranges [2].

3.2 Control Group

The control group comprised 43 students (28 males and 15 females). Similar to the experimental group, these students had an average score of approximately 73 in their prior foundational computer science course, with a distribution that spanned from lower to higher ranges, ensuring comparable initial learning levels. Having completed identical prerequisite courses, the control group had established a preliminary understanding of computer science fundamentals and demonstrated basic programming logic. This foundational alignment allowed both groups to engage with C language instruction at comparable levels, minimizing interference from varying prior knowledge. Observations revealed notable diversity in learning styles among control students. However, some exhibited greater reliance on teacher-led instruction, showing weaker initiative in self-directed exploration. Teamwork participation and collaboration skills also required improvement. These characteristics reflect traditional teaching methodologies, forming a contrast with the experimental group that facilitates subsequent analysis of different instructional approaches [3].

4. Practice Process

4.1 Experimental Class (Project-Based Learning Method)

At the start of the semester, teachers meticulously designed a series of projects based on the C language syllabus and real-world applications. These projects not only covered essential C concepts like data types, control structures, functions, arrays, and pointers but also incorporated elements of fun and practicality to spark students' interest and motivation. Three main projects were finalized: "Campus Sports Meet Score Management System," "Simple Supermarket Cashier System," and "Personal Contact List Management System." Each project underwent detailed task decomposition. Taking the "Campus Sports Meet Score Management System" as an example, it was broken down into subtasks such as athlete information entry, event setup, score recording and modification, score retrieval and statistics, and award information generation. Each subtask specified clear functional requirements and expected outputs, ensuring students had a clear understanding of their responsibilities.

As the project progresses, the teacher strategically incorporates relevant C programming knowledge. For instance, in the athlete information entry subtask of the "Campus Sports Meet Score Management System," which involves using structures, the teacher first introduces the concept of structures, explaining their definition, declaration, and initialization methods. Then, through a simple code example, the teacher demonstrates how structures can store students' basic information. Throughout the explanation, the teacher emphasizes the importance of structures in organizing and managing complex data in real-world projects.

During the project implementation, students are organized into groups of 4-5 members to develop the project. Through discussions, each group member assigns responsibilities for specific subtasks or modules. When encountering issues during implementation, students first resolve problems through group discussions. For example, when implementing the product pricing calculation function in the "Simple Supermarket Cashier System," a group might encounter data type mismatches, causing calculation errors. Members then consult textbooks, reference materials, or recall classroom knowledge to identify and solve the problem. If unresolved internally, students can seek help from teachers. During classroom patrols, teachers address common issues through class-wide discussions and provide individualized guidance for unique challenges. Additionally, teachers encourage students to explore diverse solutions, fostering innovative thinking and problem-solving skills.

When the project reaches 50% completion, the teacher conducts a mid-term review. Each group must present its progress to the class, detailing completed tasks, challenges and solutions, pending tasks, and estimated timelines. Other group members may raise questions or suggestions, while the teacher provides feedback and guidance based on their presentations. For instance, if a group struggles with the contact search feature in the "Personal Contact

Management System,” the teacher may guide them to optimize the algorithm for faster searches. Through this mid-term review, students can promptly identify shortcomings in their projects, adjust development plans, and ensure timely and high-quality completion [4].

4.2 Control Group (Traditional Teaching Method)

After each chapter’s theoretical instruction, students complete corresponding lab sessions featuring verification and foundational experiments designed around the chapter’s key concepts. For example, following an array study, students might develop a program to calculate the average of an integer array. During lab sessions, they follow instructor-provided manuals while receiving real-time guidance and troubleshooting support. Upon completion, students submit lab reports detailing objectives, procedures, code implementation, and results.

5. Comparison of Practical Results

5.1 Comparison of Academic Performance

Two classes took the final exam with standardized test design, invigilation, and grading. The exam was scored out of 100 points, with the following question types:

(1) Multiple-choice questions: 20 questions in total, each worth 2 points, for a total of 40 points. The questions primarily assess fundamental knowledge of C language, including basic concepts and syntax rules.

(2) Fill-in-the-blank questions: 10 questions in total, each worth 2 points, for a total of 20 points, focusing on memorizing and applying key knowledge points.

(3) Program comprehension questions: Comprising three code segments, each containing multiple questions, totaling 20 points. This section evaluates students’ ability to understand code logic.

(4) Programming Challenge: Two questions, each worth 10 points, totaling 20 points. Students are required to write complete C programs using their acquired knowledge, with emphasis on evaluating their practical programming skills and problem-solving abilities.

The experimental class achieved an average score of 82.5, significantly higher than the control class’s 73.8 (an 8.7-point difference). The independent samples t-test yielded a t-value of 4.56 (df=86, $p<0.001$), demonstrating a statistically significant difference between the two classes. The score distribution is as follows:

(1) 90~100 points: 12 students in the experimental class, accounting for 26.7% of the total class; 5 students in the control class, accounting for 11.6% of the total class.

(2) 80~89 points: 18 students in the experimental class, accounting for 40.0%; 13 students in the control class, accounting for 30.2%.

(3) 70~79 points: 10 students in the experimental class, accounting for 22.2%; 14 students in the control class, accounting for 32.6%.

(4) 60~69 points: 3 students in experimental class, accounting for 6.7%; 7 students in control class, accounting for 16.3%.

(5) Scores below 60: 2 students in the experimental class (4.4%); 4 students in the control class (9.3%).

The score distribution data reveal that the experimental class demonstrates a significantly higher proportion of high scorers (80 points or above) compared to the control class, while the proportion of low scorers (below 60 points) is lower in the experimental group. This indicates that the project-driven teaching method effectively enhances students’ overall academic performance, enabling more learners to achieve better educational outcomes [5-8].

5.2 Comparison of Practical Abilities

To comprehensively and objectively evaluate the practical skills of students from two classes, a specialized practical assessment was conducted after the final exams. The assessment featured a comprehensive project task titled “Small Enterprise Employee Information Management System,” which required students to complete it independently within 4 hours. The project covered core C language concepts, including structures, file operations, function calls, and linked lists, while also demanding students’ ability to design systems and solve problems. Following the assessment, a judging panel of three experienced teachers with rich teaching and practical experience scored the projects based on detailed evaluation criteria, as outlined below:

(1) Functional completeness (40 points): The system should support employee information entry, search, modification, and deletion, along with extended features like payroll calculation and department statistics. Scoring is based on the completeness of implemented functions.

(2) Code Standardization (25 points): This includes variable naming conventions, code indentation, and comment clarity. Scores are higher for code that adheres to standards and is highly readable.

(3) Algorithm Rationality (20 points): This evaluates whether the algorithms used by students to implement various functions are efficient and logical. For example, using a well-designed search algorithm in the employee information query function can lead to higher scores.

(4) Debugging and Error Correction Skills (15 points): This evaluates students' ability to quickly identify and resolve programming errors. During the assessment, students may debug their code, with scores determined by both the time spent debugging and the effectiveness of their solutions.

The practical assessment results showed that the experimental class achieved an average score of 78.6, significantly higher than the control class's 65.2 (a difference of 13.4 points). The independent samples t-test yielded a t-value of 5.89 ($df=86$, $p<0.001$), confirming a statistically significant difference between the two classes.

In the function integrity, the average score of the experimental class is 32.5, and the average score of the control class is 23.8. Most students in the experimental class can realize the main function of the system completely, and some students can realize some extended functions; while there are more students in the control class who have a function deficiency.

In terms of code standardization, the experimental class achieved an average score of 18.2, while the control class scored 13.5. Students in the experimental class demonstrated more standardized practices in variable naming and code formatting, resulting in more readable code. Conversely, some students in the control class exhibited disorganized code structures and unclear variable naming.

6. Conclusion

In conclusion, the project-driven C language teaching reform has demonstrated remarkable effectiveness. This methodology overcomes the limitations of traditional approaches, elevates students' academic performance and practical competencies, while fostering learning motivation and initiative. These outcomes offer innovative perspectives for C language instruction, which can be further refined and expanded to other courses in the future.

References

- [1] Liu H, Wang X. Interactive curriculum reform and practice research based on practical competence: a case study of "Introduction to Computational Thinking (C Language)". *J Chizhou Univ.* 2024;38(06):157-160.
- [2] Liu C, Huang H. Research and practice on teaching reform of C language course. *Sci Technol Horizon.* 2024;14(32):12-14.
- [3] Hua X, Chen Y, Chen R. Teaching reform and practice of C programming course under the new gaokao system: a case study of Taizhou University. *J Taizhou Univ.* 2024;46(03):77-81+92.
- [4] Zhang C, Li X, Zhang Y, et al. Teaching reform and practice of C language course based on task-oriented microcontroller. *High Educ J.* 2024;10(11):125-128.
- [5] Wang L, Liu F, Qiao J, et al. Teaching reform and practice of project-driven C programming course. *Comput Knowl Technol.* 2022;18(26):157-159.
- [6] Jing Z. Research on the teaching reform of computer C language under the mode of MOOC. *DEStech Trans Soc Sci Educ Hum Sci.* 2017(11):112-123.
- [7] Ting C. Research on the experiment teaching reform of "C Language Programming". *Exp Technol Manag.* 2010;27(10):182-184.
- [8] Min C. The research on the project-driven teaching reform of C language program designing in vocational colleges. *J Qingdao Tech Coll.* 2011;24(01):58-61.