International Journal of Education and Social Development
ISSN: 3078-9931 (Print), ISSN: 3078-2287 (Online) | Vol. 5, No. 3, 2025

Exploration and Practice of Teaching Reform in C Language Programming Based on the CDIO Model

Zhihui Li, Xiaoshuo Jia and Shaoze Zhang

College of Computer Science, Guangdong University of Science and Technonlogy, Dongguan 523000, China

Abstract: The cultivation of applied talents requires undergraduate students in the computer science major to possess solid practical programming skills and innovation capabilities. The success of practical teaching reforms is directly related to the development of students' engineering practice abilities. Based on the construction of undergraduate computer science course clusters and the experience and achievements of teaching reforms, this paper proposes an exploratory and practical teaching reform plan for C language programming under the guidance of the CDIO model. The plan includes reforms in theoretical teaching methods, experimental teaching methods, and assessment mechanisms, aiming to establish a practice-oriented teaching model. This reform will significantly enhance students' learning interest and cultivate their abilities in autonomous learning and practical innovation.

Keywords: CDIO Model, C Language Programming, Exploration and Practice.

1. Introduction

"Programming in C" is the foundational core course in programming for science and engineering majors in higher education institutions, such as electronic information, computer science and technology, automation, and the Internet of Things. It plays a pivotal role in bridging students from "theoretical understanding" to "engineering practice," fostering programming mindset, mastering modular programming methods, and cultivating an engineering-oriented development philosophy. The teaching effectiveness of this course not only directly affects students' acceptance of subsequent professional courses such as "Data Structures," "Operating Systems," and "Embedded System Development," but also plays a decisive role in students' adaptation to core competitiveness in engineering positions such as software development, system integration, and algorithm design after graduation. In today's rapidly developing digital economy, programming ability has become an "essential literacy" for science and engineering talents, and "C Language Programming" is an important starting point for cultivating this literacy.

However, under the traditional teaching mode, the teaching and assessment of "C Language Programming" have long had the drawbacks of "emphasizing theory over practice, emphasizing knowledge over ability". Course teaching often focuses on teaching grammar knowledge points, and classroom cases are limited to simple verification questions in textbooks, lacking connection with practical industrial application scenarios. The assessment method is mainly based on the final closed book exam, with test questions designed to focus on theoretical aspects such as grammar rule memory and keyword usage analysis. Even if programming questions are included, they are mostly isolated algorithm verifications that are detached from practical scenarios, making it difficult to comprehensively measure students' code writing standardization, problem analysis ability, system design thinking, and engineering practice literacy. This teaching and assessment model directly leads to the common phenomenon of "high scores but low abilities" among students - some students can achieve excellent results in the written test, but find it difficult to independently develop a small program with practical application value, and are at a loss when facing complex engineering problems, which is seriously disconnected from the demand of enterprises for application-oriented and innovative engineering and technical talents.

The CDIO engineering education model is an advanced engineering education concept jointly proposed by four top universities in the world, including Massachusetts Institute of Technology and Chalmers University of Technology in Sweden. Its core is to simulate the "Conceptual Design Implement Operate" process of the entire life cycle of industrial products, closely combining theoretical teaching with engineering practice, emphasizing project as the carrier and ability cultivation as the core, comprehensively enhancing students' engineering practice ability, teamwork ability, and innovative thinking. Since its proposal, this model has been widely applied in engineering majors in over a hundred universities worldwide and has achieved significant results. Integrating the CDIO model into the teaching reform of "C Language Programming", especially the reform of assessment methods, can effectively break the limitations of traditional assessment. By constructing an assessment system that is in line with engineering practice, the focus of teaching can be shifted from "knowledge imparting" to "ability cultivation", achieving the teaching goal of "promoting learning, practice, and ability through exams". Students can experience the complete engineering development process during the assessment process, gradually form engineering thinking and teamwork awareness, and better meet the requirements of talent cultivation in the new era of engineering education, laying a solid foundation for their future career development.

2. Current Situation and Existing Problems of Course Assessment

The C language programming course is a core foundational course for computer science majors, serving as the foundation for all programming directions and a prerequisite for other subsequent courses. It is also a key knowledge subject for enterprise recruitment and postgraduate entrance

examinations. C language is a course that emphasizes both theory and practice. To some extent, it can even be said that practice is more important than theory, because ultimately, the presentation of teaching results and students' learning outcomes must fall into programming practice. Therefore, experimental and practical teaching should become the most important part of C language teaching. However, currently there is a phenomenon of putting the cart before the horse in the teaching of C language in Chinese universities.

2.1. Imbalance in Course Teaching Content

Currently, most universities adopt a model of "classroom theoretical teaching as the main focus, supplemented by experimental teaching" for "C language programming", with teacher centered curriculum implementation and grammar knowledge teaching as the driving force. In classroom teaching, teachers spend a lot of time explaining theoretical knowledge such as C language grammar rules and keyword usage, and only provide confirmatory explanations through simple examples in textbooks, lacking systematic cultivation of practical abilities such as programming thinking construction, problem analysis methods, and code debugging skills. Students are always in a passive receiving state, mechanically memorizing knowledge points following the teacher's teaching rhythm, lacking space for active thinking and independent exploration, and generally have weak learning initiative. This teaching model, which emphasizes theoretical indoctrination over practical guidance, makes it difficult for teachers to grasp students' knowledge absorption in real time and discover weak links in teaching in a timely manner; Students often face the dilemma of "being able to understand but not programming", making it difficult to translate abstract grammar knowledge into practical programming skills, resulting in unsatisfactory learning outcomes.

2.2. Lack of Supervision in After-School Practice

As a basic public course, "C Language Programming" is taught in large classes in most universities, with a class size of 50-100 people. After classroom teaching, teachers are limited by time and energy and cannot effectively supervise each student's post class practice: they cannot confirm whether students have truly completed programming exercises independently, nor can they grasp the specific problems encountered by students in practice; The homework submitted by students is mostly the final result, and teachers cannot trace their problem-solving ideas and debugging process, making it difficult to judge students' true mastery level. This leads to teachers being unable to obtain effective teaching feedback in a timely manner, unable to accurately identify students' knowledge gaps and skill shortcomings, and even more difficult to carry out targeted personalized tutoring. In the end, students' after-school practice relies entirely on their own consciousness. Some students with weak foundations and lack self-discipline often give up practical exercises or resort to copying homework to cope with assessments, and the cultivation of practical abilities is merely a formality.

2.3. Experimental Topic Solidification

According to the requirements of the experimental outline, the traditional experimental questions in "C Language Programming" are mostly verification content of fixed chapter knowledge points, with a small quantity and a single form, highly similar to textbook examples and post class exercises, lacking innovation and comprehensiveness. The experimental task only requires students to complete code writing and verify syntax correctness according to established steps, without considering code readability, maintainability, or innovative design. This fixed experimental mode allows students to complete tasks by simply applying programming templates from textbooks, lacking training in the ability to analyze, design, and solve complex problems. Programming thinking and innovation skills are not effectively cultivated and improved, which is seriously disconnected from the core competency requirements of "design implementation" emphasized by the CDIO mode.

2.4. Single Assessment Method

The current assessment system for "C Language Programming" has significant deficiencies: on the one hand, the assessment method is mainly based on "final closed book exams, supplemented by regular grades", and regular grades rely heavily on attendance and homework submission, lacking comprehensive monitoring of the learning process; The final exam focuses on testing theoretical content such as grammar knowledge memory and fixed question answering, but lacks examination of programming practical ability and problem-solving ability. On the other hand, the assessment content is limited to the verification of a single knowledge point, lacking comprehensive questions that are combined with actual engineering scenarios; The evaluation criteria are fixed in a "standard answer format", which lacks tolerance for students' personalized programming ideas and innovative algorithm designs. This assessment model, which emphasizes results over processes and theory over practice, guides students to focus too much on their final written test scores and neglect their daily programming exercises and innovative attempts, resulting in ineffective training of their engineering and practical abilities, making it difficult to achieve the talent cultivation goals of the course.

3. Reform Measures for Course Assessment Methods

The fundamental core of the reform of the C language programming course is guided by the CDIO model and constructivist theory, transforming the teaching plan of "theoretical lectures as the main focus" into a teaching model of "experimental practice leading theory, practice emphasizing theory" and "learning by doing" as the core. The reform emphasizes the integration of practice throughout the teaching process, adopting a task driven teaching approach that puts students at the center and builds a positive and harmonious learning environment for them. Discussion based learning and research-based learning are the main methods, and practical cases are used to guide students to independently construct a complete knowledge system. The main reform measures include the following aspects.

3.1. Optimize Teaching Content and Mode

Break the teaching logic of "grammar teaching as the core" and build a teaching system of "theoretical knowledge practical cases project driven". In classroom teaching, grammar knowledge is integrated into practical application scenarios, and teaching cases are designed with a "problem oriented" approach (such as simple calculator development,

student information statistics, etc.). Through the complete process of "raising questions, analyzing ideas, designing solutions, writing code, debugging and optimizing", theoretical knowledge and practical methods are explained simultaneously. Reduce pure theoretical teaching time, increase classroom interactive programming, group discussions, case analysis and other activities, and guide students to actively participate in the knowledge building process. According to the principles of constructivist theory, we have designed a "three-step teaching model" for each lesson that is led by programming practice, consisting of "pre class preview, in class discussion, and post class practice". All steps follow the principle of "practice first, theory later, and innovation later", while also adhering to the four steps of the CDIO model. This cycle runs through the entire teaching process.

3.2. Strengthen Process Supervision and Hierarchical Training

Utilizing online tools such as the PTA Programming Design Teaching Platform and LeetCode Education Edition, a tiered after-class practice system is established: weekly assignments include three levels of programming tasks—"Basic Problems (to reinforce knowledge points) + Advanced Problems (to enhance application) + Extended Problems (to stimulate innovation)." Students are required to complete and submit these tasks within the specified time. The platform records real-time data on students' code submission frequency, debugging process, problem-solving duration, and code execution results. Teachers analyze this backend data to accurately identify students who fail to participate, complete tasks perfunctorily, or encounter significant difficulties. For common issues, concentrated explanations are provided in the next class; for individual problems, timely feedback is delivered via platform messages, online Q&A groups, or one-on-one tutoring to ensure targeted and prompt assistance.

3.3. Innovative Topic Design

Refactoring experimental teaching content, reducing the proportion of confirmatory experiments, and increasing comprehensive, design oriented, and innovative experimental topics. Select representative practical cases from real exams such as computer proficiency tests and Blue Bridge Cup competitions, or break down large tasks from school enterprise cooperation projects and extract appropriate cases to solve practical problems. Effectively solve the problem of case functions being too single, not innovative and practical enough, while promoting the application of learning and facilitating the transformation of students' learning outcomes. At the same time, it also promotes the implementation of the national "new engineering" construction concept, combining learning with application.

3.4. Reform the Assessment System

Refactoring the proportion of assessment scores, combining process based assessment (50%), project-based assessment (30%), and summative assessment (20%), to achieve the assessment orientation of "emphasizing process, strengthening practice, and promoting innovation". The comprehensive score composition and proportion are shown in Table 2.

3.4.1. Process Assessment

Process assessment covers three aspects: ① Classroom performance (10%): Scoring based on student classroom interaction, group discussion participation, and completion of programming cases; ② Post class programming training (20%): Based on online platform data, comprehensive scores are given from dimensions such as code correctness, running efficiency, readability, and timely submission; ③ Periodic testing (20%): Conduct a computer-based test every 4 weeks, focusing on testing students' comprehensive application ability of recent knowledge points and adjusting teaching strategies in a timely manner.

3.4.2. Project Based Assessment

Taking the CDIO "conception design implementation operation" lifecycle as the clue, students are required to complete a small-scale engineering project with practical application value as a team (3-4 people). Students need to go through four stages: requirement analysis, scheme design, code implementation, and functional testing. They need to submit requirement analysis reports, system design documents, source code, testing reports, and project demonstration videos. Assessment is evaluated from three dimensions: process (project progress, division of labor and collaboration), outcome (functional integrity, technical rationality, document standardization), and team collaboration (member contribution, communication efficiency). The member contribution is determined through a combination of "teacher evaluation-member peer evaluation".

3.4.3. Summative Assessment

The final exam adopts a combination of computer-based testing and defense: the computer-based testing section designs comprehensive programming questions to test students' abilities in problem analysis, code writing, and debugging; The defense section requires students to elaborate on the design ideas, technical difficulties, and solutions of the final project, and to test their logical expression and innovative thinking. The evaluation criteria adopt a "basic score-optimization score-innovation score" model, encouraging students to use novel algorithms, concise code, or expanded functions, breaking the constraints of "standard answers".

Table 1. Composition and Proportion of Comprehensive Grades

assessment category	Process assessment results (50%)			Project based assessment (30%)	Summative assessment (20%)	
assessment items	Classroom performance	post class programming training	phased testing	project completion	computer-based exam (final exam)	defense
assessment ratio	10%	20%	20%	30%	15%	5%

4. Implementation Effect

The implementation of the assessment mode reform in this course has significantly enhanced students' learning initiative through problem oriented teaching, layered practical tasks, and process based assessment incentives. After the reform,

the average submission rate of students' online programming tasks increased from 75% to 95%, the classroom interaction participation rate increased from 60% to 88%, and the proportion of students who independently searched programming materials and participated in online programming community discussions increased significantly, forming a good learning atmosphere of "active thinking and willingness to practice". After grading the final exam papers and summarizing the data, the students' final paper scores and overall evaluation scores are obtained, and the final paper scores and overall evaluation scores are analyzed.

4.1. Analysis of Final Exam Scores

Through data statistics, the percentage table of students in each score range and the distribution chart of students' final exam scores were obtained, as shown in Table 2 and Figure 1, respectively.

Table 2. Percentage of Students in Each Score Segment of the Final Exam Paper

Score range	<60	60-69	70-79	80-89	90-100
Percentage	7.1%	20.1%	37.4%	27.8%	7.6%

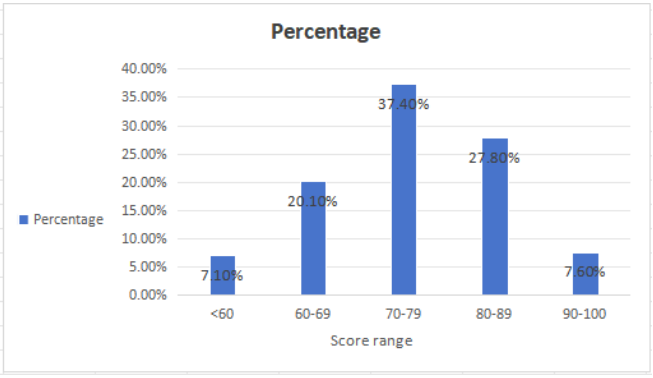

Figure 1. Distribution of Students' Final Exam Scores

From Figure 1 and Table 2, it can be seen that the overall distribution of students' final exam scores is normal, with a pass rate of 88.9%, mainly concentrated in the 60-89 segment. The test questions are in line with the actual teaching situation, with moderate difficulty and good differentiation. The overall learning situation of the students is good, and their understanding and mastery of the knowledge points are in place. However, the number of high scoring students above 90 points is relatively small, indicating that the students' comprehensive application ability of the knowledge points still needs to be improved.

4.2. Analysis of Overall Evaluation Score

The overall evaluation score consists of process assessment (70%), project-based assessment (30%), and summative assessment (20%). By summarizing and summarizing the scores obtained from various process assessments, a percentage table of the number of students in each score range and a distribution chart of the overall evaluation score are obtained, as shown in Table 3 and Figure 2, respectively.

Table 3. Percentage of Students in Each Score Segment of the Overall Student Evaluation Score

Score range	<60	60-69	70-79	80-89	90-100
Percentage	1.4%	5.6%	30.5%	43.1%	19.4%

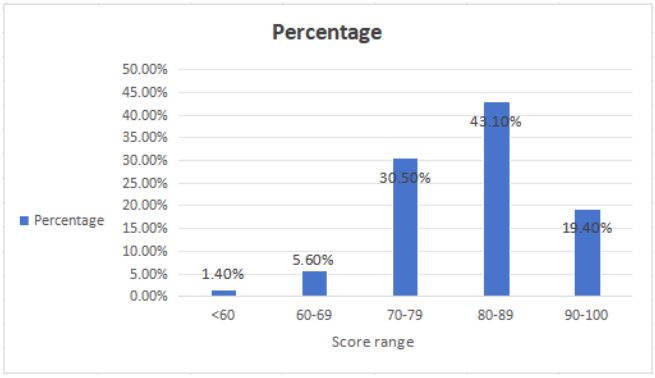

Figure 2. Distribution of Overall Student Evaluation Scores

According to Figure 2 and Table 3, the overall distribution of students' overall evaluation scores is normal, with a pass rate of up to 98.6%, mainly concentrated in the 70-100 segment. A very small number of students failed the overall evaluation due to their lack of active participation in process teaching activities, indicating a high degree of achievement of course objectives. The diversified assessment system more comprehensively reflects students' learning situation and ability level, effectively avoiding the phenomenon of "high scores but low abilities". After the reform, the matching degree between students' grades and actual abilities has significantly improved. At the same time, the introduction of process assessments and online platform data has reduced the randomness caused by a single exam, improved the fairness and objectivity of assessments, and gained widespread recognition from students and peers.

5. Summary

The teaching reform of "C Language Programming" based on CDIO mode has systematically solved the core drawbacks of traditional teaching, which emphasizes theory over practice, knowledge over ability, and results over process, by optimizing teaching content, improving practical system, and innovating assessment methods. Teaching practice has shown that this reform plan can fully mobilize students' learning initiative, comprehensively enhance their programming practice ability, engineering literacy, innovative thinking, and teamwork ability, and significantly improve teaching quality.

Acknowledgements

This work was financially supported by GKZLGC2023034, GKZLGC2024153, GKZLGC2024180 fund.

References

[1] Liu, Y., Li, H. CDIO-Based Formative Assessment System for CProgramming Courses: Designand Practice. Journal of Engineering Education Transformations. 2022, Vol.35(No.2), p.47-59.

[2] Zhang, Q., Wang, L., Chen, J. Project-Driven Learningin CProgramming: ACDIO-Oriented Reformto Enhance Practical Competence. IEEE Transactionson Education. 2023, Vol.66(No.3), p.189-196.

[3] Ahmed, M., Smith, K. Addressing "High Scores but Low Ability" in CProgramming: ACDIO-Integrated Assessment Approach. European Journal of Engineering Education. 2021, Vol.46(No.4), p.512-528.

[4] Lee, S., Park, J., Kim, H. Online Programming Platformsin CDIO-Based CLanguage Courses: Enhancing Supervision and

Feedback Efficiency. Computer Applications in Engineering Education. 2024, Vol.32(No.1), p.215-231.

[5] Garcia, R., Martinez, A. Innovating Assessment Criteria for CProgramming: Flexibilityand Creativity in CDIO Framework. Journal of Computing Sciences in Colleges. 2022, Vol.38(No.3), p.78-87.

[6] Wang, H., Zhang, Y. Integrating CDIO with Blended Learning for CProgramming: Improving Student Engagement and Practical Skills. Education and Information Technologies. 2021, Vol.26(No.6), p.6891-6910.

[7] Davis, E., Brown, S. CDIO-Oriented Reform of Experimental Teaching for CProgramming: From Verification to Innovation. Journal of Engineering Education. 2025, Vol.114(No.1), p.89-105.

[8] Chen, L., Yang, X., Hu, Z. Team Collaboration Assessment in Introductory CProgramming Courses: ACDIO Perspective. Proceedings of the International Conferenceon Engineering Education (ICEE). Singapore, 15-17June2023, p.145-150.