Curriculum and Teaching Methodology (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/curtm.2024.070507
ISSN 2616-2261 Vol. 7 Num. 5

Research on C Language Programming Case-Assisted Teaching Based on BUGs Exclusion

Yuanzi He[a], Yantao He[b,*]

Department of Computing, Guangdong University of Science and Technology, Dongguan, China
[a]heyuanzi@gdust.edu.cn, [b]heyantao@gdust.edu.cn
[*]Corresponding author

Keywords: C Language Programming, Case-Assisted Teaching, BUGs Exclusion

Abstract: This paper seeks to thoroughly implement the pedagogical philosophy of "student-centered learning, teaching students according to their aptitude, and categorizing training." During the project's execution, we will use the core requirements of the C Language Programming Design course as our benchmark. We will meticulously design and transform a series of practical cases that are closely related to real-life scenarios. These cases not only possess high practicality and interest but also cleverly incorporate specific small errors (BUGs), thereby providing students with a genuine programming environment. This paper will offer robust support and assurance for the transformation of the C Language Programming Design course from knowledge-intensive classrooms to ability-based ones. By introducing practical cases and engaging in BUGs elimination practice activities, we aim to assist students in transitioning from theoretical knowledge to practical application. This will foster their innovative abilities and team spirit, thereby laying a solid foundation for their future career development and lifelong learning.

1. Introduction

In contemporary higher education, "C Language Programming" is a foundational course for computer science majors[1]. The quality of instruction in this course is paramount to the development and enhancement of students' programming skills. This course serves not only as an introduction to the world of programming but also as a crucial bridge for students to establish a robust programming foundation and comprehend the principles of computer science. Initially, the "C Language Programming" course offers students a comprehensive and systematic introduction to programming. For novices, programming can be daunting due to its inherent complexities. However, the C language, recognized as a classic and efficient programming language, possesses a clear syntax structure and powerful features that enable students to quickly grasp the fundamental concepts and methods of programming. Through studying C language, students can understand the basic structure of programs, the definition and usage of variables, the application of control statements, the definition and calling of functions, and other core programming knowledge, thereby laying a solid foundation for subsequent learning of other programming languages. Secondly, the "C Language Programming" course fosters students' logical thinking and problem-solving abilities. Programming is not merely a skill but also a mode of thinking. In the process of writing programs,

students are required to use logical thinking to analyze and solve problems, and implement program functions through algorithm design. The development of logical thinking through the study of C language is crucial for students' future academic and professional pursuits. The acquisition of this skill not only enhances their problem-solving abilities but also equips them to navigate complex and dynamic programming challenges. Furthermore, the course "C Language Programming Design" plays a pivotal role in shaping the career trajectories of computer science majors. In the contemporary information age, where computer technology permeates every aspect of our lives, there is a growing demand for professionals skilled in this domain. Given that the C language stands as a cornerstone in computer science, with applications spanning system software, embedded systems, game development, and beyond, mastery of its programming nuances is paramount for the career advancement of computer science majors. The acquisition of C language proficiency not only expands students' employment opportunities but also enhances their competitiveness, thereby establishing a robust foundation for future career progression. Furthermore, the "C Language Programming" course is instrumental in advancing discipline construction and fostering teaching reform. As computer technology continues to evolve, new programming languages and technologies emerge, presenting fresh challenges to the traditional "C Language Programming" curriculum. To adapt to these changes, educators must consistently update their teaching content and methodologies, incorporate novel teaching concepts and technical tools, and strive to improve the course's teaching quality and effectiveness. Concurrently, by emphasizing practical teaching components and enhancing students' practical skills and innovative spirit, there is a promotion of ongoing deepening in both discipline construction and teaching reform.

In addressing the aforementioned issues, numerous experts have offered their perspectives. The study conducted by Dema[2] offers insights into the diverse preferences and methodologies adopted by students in learning C programming. A comprehensive survey was carried out to examine student learning habits, cognitive styles, and motivations. The findings reveal several distinct learning styles, namely visual, auditory, and kinesthetic learners. Visual learners predominantly acquire knowledge through reading code and comments, as well as observing diagrams and flowcharts. They favor structured learning materials and comprehend concepts by examining examples. Conversely, auditory learners prefer learning through lectures, discussions, and verbal explanations. They enhance their understanding of programming concepts by listening to explanations and engaging in dialogue. Kinesthetic learners, on the other hand, learn by writing code, solving problems, and completing projects. They reinforce theoretical knowledge through practical activities and discover and solve problems in real-world scenarios. Daungcharone[3] investigates the efficacy of converting mobile games into a Lecture-Based Approach (LBA) to enhance C language education. Through the creation of interactive and engaging mobile games, researchers incorporated essential concepts and programming skills of the C language into game scenarios. This approach allowed students to acquire programming knowledge while enjoying themselves. Experimental results suggest that, compared to traditional teaching methods, this LBA transformed from mobile games significantly boosts student motivation and satisfaction in learning. The primary focus of He[4] is the integration of C language instruction with its practical application in specific professional fields, utilizing innovative teaching methodologies to enhance students' professional skills and problem-solving abilities. Traditional C language education has often been criticized for its emphasis on theoretical knowledge, neglecting the connection with professional practice. To address this shortcoming, this paper proposes a series of teaching reform measures, including case teaching, project-driven learning, and interdisciplinary cooperation. By incorporating real-world cases and projects relevant to students' majors, students can not only learn and apply C language programming skills in a practical context but also deepen their understanding of professional knowledge. Wan[5] detailed the creation and implementation of an innovative tool for

teaching C language programming: interactive virtual algorithm animation. This tool augments students' comprehension of abstract concepts by visually representing the algorithm execution process. In a pedagogical context, this animation not only aids in classroom explanations but also functions as a self-learning resource and experimental platform, thereby effectively enhancing students' learning efficiency and engagement. Research has demonstrated that this novel teaching method significantly contributes to the promotion of algorithm learning. Malysheva[6]investigates an innovative pedagogical support strategy designed to anticipate student needs based on the specific error types observed in their programming assignments. The objective is to create an automated system capable of pinpointing prevalent error trends in student code, subsequently utilizing this data to ascertain whether students are grappling with comprehension or mastery challenges. This would enable the provision of timely and personalized instructional assistance.

This paper concentrates on fostering computational thinking with the objective of augmenting students' application skills in C language. Particular emphasis is placed on enhancing the performance of students who struggle with programming. Through practical development techniques such as code reading, completion, and BUGs debugging, students are directed to execute case programs, identify and resolve issues. This approach enables them to gain a profound understanding of coding logic through the "try-modify-improve" process. Concurrently, this paper thoroughly enacts the teaching philosophy of "student-centered, teaching according to aptitude, and categorized training." It designs and modifies practice cases that are closely related to real-life scenarios based on the requirements of the "C Language Programming Design" course. By integrating BUGs elements, it creates a genuine programming environment. The aim is to facilitate the transition of the course from being knowledge-intensive to ability-based, cultivate students' innovative abilities and team spirit, and establish a solid foundation for their future career progression and lifelong learning.

2. The Drawbacks of Traditional Teaching Methods

2.1. Larger Student Size, Greater Teaching Difficulty

The exponential growth in college enrollment has led to a significant increase in the number of students enrolled in the "C Language Programming" course. This surge not only necessitates expanded educational provisions but also presents unprecedented challenges to the teaching methodology. Given this context, it becomes challenging for educators to provide individualized attention and guidance to each student. The growing disparities in student abilities further complicate matters, as traditional teaching methods struggle to cater to the diverse needs of all students. This directly impacts the quality of instruction and the learning outcomes of the students. To mitigate these issues, some institutions have begun to explore alternative teaching models such as small-class instruction, tiered teaching, or online teaching. However, these innovative models come with their own set of limitations, including high resource requirements and stringent demands on faculty expertise. Consequently, finding an effective strategy to manage the continuous expansion of student numbers while maintaining teaching quality has become a critical topic in the realm of college education reform.

2.2. The Wide Range of Student Bases Makes it Difficult to Standardize the Pace of Instruction

In the instruction of the "C Language Programming" course, the disparities in students' computer foundational knowledge and learning capacities have emerged as a significant challenge. These variations result in uneven progress and mastery among students, thereby complicating the

scheduling of teaching activities for educators. On one hand, teachers may need to decelerate their teaching pace and repeatedly elucidate fundamental concepts to accommodate students with weaker foundations. On the other hand, such a pacing could potentially induce boredom and lack of challenge among students with more robust foundations. Consequently, striking a balance between maintaining teaching quality and catering to diverse student needs has become an intricate problem for educators to address.

2.3. Inadequate Construction of Curriculum Resources and Lack of Teaching Resources

In certain universities, the development of course resources for "C Language Programming" has been insufficiently prioritized and funded, leading to a notable scarcity of instructional materials. This deficiency poses significant challenges to educators, hindering their ability to source appropriate teaching aids and case studies. Such limitations not only constrain the pedagogical approaches teachers can employ but also impede students' comprehension and mastery of the subject matter. The absence of engaging case analyses often makes it challenging for students to integrate theoretical concepts with real-world scenarios, potentially diminishing their motivation and learning efficacy. Furthermore, the dearth of instructional resources contributes to the uniformity of teaching content. Without a variety of resources, educators struggle to enrich and deepen the curriculum, resulting in a narrower exposure of students to the subject and failing to satisfy their knowledge needs.

2.4. Difficult for Students to Digest and Absorb

The content of the C language programming course is both complex and extensive, often necessitating an extended period to ensure comprehensive teaching. However, in certain institutions, due to various factors, this course is allocated a relatively high number of class hours. This density can pose challenges for students, as they struggle to fully comprehend and internalize the knowledge within the limited timeframe. The pressure to learn intensifies under these circumstances, as students are required to assimilate a vast amount of information and knowledge rapidly. This rapid pace may hinder their ability to effectively organize and categorize the learned material, thereby affecting their mastery and application of the knowledge. Furthermore, the accelerated teaching rhythm may deprive students of sufficient time for practical application and exercise. Given that C language programming design is a highly practical course, students who lack hands-on experience often find it challenging to deeply understand the concepts and skills of program design. This deficiency can potentially impede their future learning and professional development.

2.5. Examination Methods are Outdated and it is Difficult to Comprehensively Evaluate Students' Abilities

In evaluating the "C Language Programming" course, some universities persist in utilizing traditional written examinations. However, given the rapid advancement of computer technology and the growing emphasis on programming practice, this conventional evaluation method is increasingly demonstrating its shortcomings. It fails to accurately and comprehensively assess students' programming and practical application skills. The traditional examination primarily focuses on theoretical knowledge, neglecting to test students' practical operation and problem-solving abilities. Students may be able to answer questions about C language programming on the test paper, but their ability to write and debug programs in practice may not be reflected in their scores. This discrepancy can lead to high scores and low abilities. Furthermore, the traditional examination does not evaluate students' innovative thinking and problem-solving abilities, which

are crucial in programming practice. To comprehensively and scientifically evaluate students' ability levels, it is necessary to reform and explore new examination methods for the "C Language Programming" course. Some universities have begun to implement more scientific and reasonable examination methods, such as computer-based examinations, practical operation assessments, and project development. These new examination methods not only assess students' theoretical knowledge but also test their practical operation ability and problem-solving ability, thereby providing a more comprehensive evaluation of students' ability levels.

3. Content Teaching for C Language Programming Case Assistance Based on BUGs Exclusion

3.1. Virtual Practice Teaching Course Group Model

The utilization of a virtual practice teaching course group mode, an innovative teaching methodology that employs virtual reality technology to emulate the genuine programming environment, offers students an immersive learning experience. Under this framework, the "C Language Programming" course is segmented into multiple virtual practice teaching groups, each overseen by a dedicated experimental teacher. These educators typically possess extensive programming and teaching experience, enabling them to provide students with immediate guidance and feedback. As students engage in programming within the virtual environment, they are able to write and debug code as they would in a real-world setting, and observe the program's running results in real time. This simulated practice not only heightens students' interest in learning but also facilitates a better understanding of abstract programming concepts. The virtual practice teaching course group mode provides students with a risk-free experimental environment by constructing simulated laboratories and programming environments. Here, students can freely experiment with various programming methods without impacting the actual system, even if errors occur. This environment fosters exploratory learning and cultivates students' innovative thinking and problem-solving abilities. In virtual practice, students can also repeatedly practice specific programming tasks, which aids in consolidating their knowledge and enhancing their skill level. Repetitive practice enables students to learn how to recover from mistakes through continuous attempts, a skill traditional teaching methods struggle to impart. Virtual practice teaching affords students the opportunity to self-regulate their learning progress. Within this virtual milieu, students are empowered to determine their own learning schedule and content, thereby significantly augmenting the flexibility and personalization of their educational experience. This adaptive learning methodology is particularly crucial for novice programmers, as it facilitates a gradual accumulation of confidence and skill. In the context of virtual practice teaching, educators assume the roles of tutors and facilitators. They not only offer immediate feedback but also scrutinize the students' learning trajectory to ensure that they remain on track. Through this interactive process, educators gain a deeper understanding of students' needs and can consequently refine their teaching strategies.

To enhance the integration of theory and practice, it is crucial to distinctly separate theoretical from practical instruction. Under this innovative pedagogical approach, teachers with a robust theoretical foundation are responsible for imparting fundamental grammar, algorithms, and programming concepts of the C language. These theoretical educators typically possess extensive academic research backgrounds and teaching experience, enabling them to elucidate complex theoretical concepts in an accessible manner and facilitate student comprehension through illustrative examples. Within the theoretical class, instructors also encourage students to contemplate the logic and principles underpinning programming, thereby fostering their abstract thinking skills. Conversely, practical instruction places greater emphasis on skill development and the application of knowledge. Practical educators are typically senior programmers with hands-on

project experience, intimately familiar with the latest industry trends and real-world requirements. During the practical course, students engage in programming exercises within virtual environments that replicate authentic programming scenarios, allowing them to apply their learned knowledge in simulated contexts. Practical educators introduce practical problems and guide students in solving them through programming, thereby deepening their understanding and application of theoretical knowledge. This separation of instruction enables students to focus on distinct learning objectives at different stages and within varied environments. Theoretical instruction establishes a solid knowledge base, while practical instruction facilitates the transformation of theoretical knowledge into practical skills. This approach effectively mitigates the disconnect between theory and practice often observed in traditional teaching, enabling students to develop their theoretical literacy and practical abilities more evenly.

3.2. BUGs' Injection and Exclusion

In the realm of C language programming instruction, the BUGs injection teaching method is increasingly gaining traction among educators. This innovative approach involves intentionally incorporating "loopholes" or errors into program code to spark students' curiosity and drive for knowledge. Such an approach encourages students to engage in active analysis and problem-solving. Consequently, students not only acquire practical programming skills but also develop the capacity to address real-world challenges.

The BUGs injection teaching method's application in C language programming education encompasses four primary steps: (1) Designing BUGs. In the execution of this teaching approach, educators must meticulously craft a series of BUGs that align with course content and student progress. These BUGs should encapsulate all facets of C language programming, including syntax, logic, and runtime errors. By employing thoughtfully designed BUGs, educators can holistically assess students' grasp of C language programming concepts while fostering their enthusiasm for active problem analysis and resolution. When devising these BUGs, educators should consider the following criteria: Firstly, BUGs should be representative, mirroring prevalent types of errors in C language programming; secondly, they should pose challenges, sparking students' curiosity and desire to explore; and finally, they should exhibit a hierarchical structure, offering varying difficulty levels to accommodate students at different stages, thereby catering to personalized instructional needs. (2) Student engagement. The successful implementation of the BUGs injection teaching method hinges on student participation. Students are required to diligently review the program code containing BUGs, attempt to compile and execute the program to identify the errors, and then utilize their acquired C language programming knowledge to debug and modify the program, eliminating the BUGs and restoring its normal functionality.(3) Interactive Discussion. The implementation of the BUGs injection teaching method necessitates an interactive discussion component. Students will inevitably encounter various challenges and difficulties during the BUGs-fixing process, which they can discuss or collaborate on to share their solutions and learned lessons. In these discussions, teachers should act as guides and facilitators, prompting students' thinking and desire for exploration through questioning and discussion guidance. Simultaneously, they can comment on and guide students' solutions, thereby aiding them in mastering programming knowledge and skills more effectively. (4) Feedback and Summary. The feedback and summary components are crucial in the implementation of the BUGs injection teaching method. Upon completion of the BUGs fix, teachers should evaluate and provide feedback on the students' work. By assessing students' solutions, code quality, problem-solving abilities, etc., teachers can gain insight into students' learning progress and existing issues, allowing them to adjust their teaching strategies and methods accordingly. Concurrently, teachers should encourage students to conduct

self-reflection and summarization. This allows students to review their performance and lessons learned during the BUGs-fixing process, consider their shortcomings and progress in programming knowledge and skills, and formulate a plan for future learning goals. Such self-reflection can help students better understand themselves, improve their skills, and establish a solid foundation for future learning and development. This approach enables them to gain a profound understanding of coding logic through the "try-modify-improve" process.

To effectively implement the BUGs injection teaching method in C language programming design instruction, educators must devise specific strategies. Initially, they should meticulously craft a series of representative and challenging BUGs that align with the students' abilities and course objectives. These BUGs should encompass all facets of C language programming, such as syntax errors, logical errors, and runtime issues. For instance, when addressing the "loop" section, educators might introduce issues like missing semicolons in for loops, incorrect condition expressions in while loops, erroneous loop terminations, and accessing out-of-bound array elements. To facilitate students' BUGs repair and debugging endeavors, educators should offer pertinent tools and resources, including debuggers, code review software, and online tutorials. Furthermore, fostering classroom interaction and discourse is crucial; educators should encourage students to share their solutions and insights. Such interactions not only enhance communication among students but also kindle their passion for programming.

3.3. Opening of Laboratories

The open laboratory model represents a pioneering approach to the allocation of teaching resources. It offers students an enhanced, flexible, and unrestricted practical learning environment by extending laboratory opening hours, augmenting experimental resources, and refining management mechanisms. Schools are compelled to devise and implement a comprehensive operation and maintenance management system to guarantee the consistent functioning of open laboratories. This encompasses establishing rules for laboratory usage, appointment protocols, safety standards, among others, and ensuring that every student has access to a fundamental computer environment within the laboratory for programming practice. In tandem with open laboratories, schools should establish a series of online practical courses. These courses can be delivered through virtual laboratory platforms, enabling students to preview and review experiments at any time and from any location, thereby significantly enhancing the flexibility and efficiency of learning. Offline open laboratories offer students a tangible operating environment where they can manipulate equipment hands-on and execute various experimental operations. This aids in their better understanding of theoretical knowledge and practice of skills. Relevant teachers are designated to maintain and manage the laboratory, ensuring that the equipment functions optimally and the environment is clean and orderly. These teachers are also expected to have the capacity to support open laboratories, respond promptly to students' queries, and guide them in experimental operations. In open laboratories, students have the opportunity to engage in hands-on programming, facilitating the transition from theoretical knowledge to practical skills. This environment also enables them to identify and address real-world problems. The flexible scheduling and abundant resources of open laboratories enable students to select practical projects that align with their learning progress and interests. This personalized approach stimulates student engagement and enhances learning efficiency. During the selection process, students are encouraged to apply their acquired knowledge to solve practical problems, a process that fosters innovative thinking and problem-solving skills. Open laboratories provide an optimal setting for teacher-student interaction, allowing for one-on-one tutoring while promoting collaborative learning among students. This interaction not only aids in the mastery of knowledge but also cultivates teamwork skills.

4. Conclusion

This paper explores the optimization of C language programming courses through the implementation of a BUG injection teaching method. This approach encourages students to engage in code reading, completion, and BUG debugging, among other practical development techniques. The goal is to facilitate the transition from theoretical knowledge to practical application, thereby enhancing students' C language application skills and computational thinking. It is noteworthy that this method is particularly beneficial for students who struggle with programming. By employing an error injection teaching method and designing practice cases that closely align with real-world scenarios, we propose a virtual practice teaching course group mode and an open laboratory. Our commitment is to promote the transformation of C language programming courses from knowledge-intensive to ability-based classrooms. This method not only enhances students' programming skills but also stimulates their learning interest and enthusiasm, thereby laying a solid foundation for their future academic and professional development.

Acknowledgements

This work is supported by the Quality Engineering Project of Guangdong University of Science and Technology under Grant GKZLGC2022038.

References

[1] Cheah C S. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review [J]. Contemporary Educational Technology, 2020, 12(2): ep272.
[2] Dema K .Understanding Students' C language programming Learning Styles: A Case Study in College of Science and Technology [J].Journal of Information Engineering and Applications, 2021, 11:7-14.
[3] Daungcharone K, Panjaburee P, Thongkoo K. Implementation of mobile game-transformed lecture-based approach to promoting C language programming learning[J]. International Journal of Mobile Learning and Organisation, 2020, 14(2): 236-254.
[4] He X, Song Y, Du Y,et al. Research on Teaching Reform of C language programming Course for Deep Integration of Professional Fields[J].Open Journal of Social Sciences, 2022, 10:267-275.
[5] Wan L, Cao Y, Shi L, et al. Development and Teaching Application of Interactive Virtual Algorithm Animation of C Language Program [J].Computational intelligence and neuroscience, 2022, 2022:7082914.
[6] Malysheva Y, Kelleher C. Using BUGs in Student Code to Predict Need for Help[C]//2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, 2020: 1-6.