Journal of Education and Educational Research
ISSN: 2957-9465 | Vol. 10, No. 2, 2024

Loop Structure Programming in C based on the UbD Model

Xiangmeng Ren [*], Hanwei Mao, Peng Wang, Liming Zhou, Zhiren Zhu

School of Electronic Engineering, Tianjin University of Technology and Education, Tianjin, China
[*] **Corresponding author**: Xiangmeng Ren (Email: 0422231001@tute.edu.cn)

Abstract: This paper discusses applying the UbD (Understanding by Design) based model in teaching C language cyclic structure program design. Aiming at the problems existing in the traditional teaching of C language, such as emphasizing theory over practice, single case, and single teaching method, the UbD model, through the concept of reverse instructional design, first determines the expected learning outcomes, and then designs the assessment evidence and specific teaching activities. The article describes in detail the three stages in the instructional design of the cyclic structure of C language using the UbD model: determining the expected outcomes, identifying suitable assessment evidence, and designing learning experiences and teaching activities. Teaching practice shows that instructional design utilizing the UbD model has significant advantages in enhancing students' interest in learning, strengthening classroom interaction, and promoting deep learning.

Keywords: UbD Theory; Reverse Instructional Design; Unit Instruction.

1. Introduction

In traditional C language teaching, there are some typical misunderstandings, such as the teaching content is too much emphasis on the theory of grammar, ignoring the practical application, resulting in students mastering the grammar but having difficulty solving practical problems; the classroom case is single, lack of challenge, and difficult to stimulate students' interest and creativity. At the same time, the single teaching method, mainly filler, lacks students' active thinking and exploration opportunities, so classroom interaction is insufficient, affecting the learning effect. The arrangement of the course duration will also impose certain limitations on teachers' teaching, and the inadequate depth of basic theory teaching will affect students' real application of C language to write programs.

Comparatively speaking, UbD (Understanding by Design), i.e. “promoting understanding by design”, advocates understanding as the priority, and has some outstanding advantages. The core idea of UbD is reverse instructional design, which first determines the teaching objectives and expected outcomes, and then determines the evaluation criteria based on the objectives and outcomes that conduct the instructional design. Adopting the UBD model for curriculum design is not only in line with the direction of the new curriculum reform but also has obvious advantages in cultivating high-quality IT talents and accurately meeting the urgent needs of society [1].

2. Unit Instructional Design based on UbD Theory

2.1. UbD Theory Overview

The UbD (Understanding by Design) theory, also known as "Teaching Design for Understanding" or "Understanding-Based Teaching Design," was first introduced by American scholars Grant Wiggins and Jay McTighe in 1998. This instructional design theory emphasizes starting with students' deep understanding and practical application skills, advocating for "backward design" and "assessment first," hence it is also referred to as "reverse engineering." The UbD instructional design model primarily consists of three stages: defining desired results, determining assessment evidence, and designing learning experiences and instruction [2, 3].

Stage 1: Determining Expected Learning Outcomes

Teachers first need to develop learning objectives that students should achieve based on the curriculum standards, the content of the textbook, and the actual situation of the students, to ensure that they are both achievable and challenging. In setting learning objectives, teachers also need to identify the “big concepts” of the unit or course, i.e., the core concepts that students need to understand in depth. At the same time, they need to set long-term transfer objectives that will enable students to effectively apply what they have learned in new contexts once they have mastered these big ideas.

Stage 2: Identifying appropriate assessment evidence

In the second stage of UbD theory, teachers need to design a series of performance tasks to assess whether students have achieved the expected learning goals. These tasks should provide a true reflection of students' understanding and application. It is also important to specify the assessment criteria, i.e. the standards that students need to meet to be considered as having mastered what they have learned. These criteria should be specific, measurable, and closely related to the learning objectives. To get a full picture of whether students have achieved the learning objectives, teachers can use other forms of assessment besides expressive tasks, such as classroom observation, group discussion, peer assessment, and so on.

Stage 3: Designing Learning Experiences and Teaching Activities

In the last stage, teachers need to design specific instructional activities based on expected outcomes and assessment evidence. These activities should be able to help students achieve the learning objectives and develop their comprehension and application skills. For students to achieve the intended learning objectives, teachers need to integrate a variety of instructional resources, such as textbooks, multimedia materials, and laboratory equipment, to support

students' learning experiences. In their teaching activities, teachers should focus on promoting deeper learning and encouraging students to engage in activities such as critical thinking, problem-solving, and collaborative communication.

2.2. Applicability of UbD Theory in Instructional Design

The essence of this model lies in the concept of "reverse design", i.e., to define the expectations of learning outcomes (i.e., "understanding") before designing teaching activities and assessment methods accordingly. In the context of teaching choice structures in C, this means that the teacher first identifies the deeper understanding goals that students need to achieve, such as being able to apply if-else statements flexibly to solve real-world problems, and understanding the decision-making process behind conditional logic, etc. The teacher then designs challenging activities around these goals. Subsequently, challenging and practical projects and cases are designed around these goals, allowing students to actively explore and think deeply in the process of problem-solving.

The UbD model not only helps students acquire a solid grasp of C syntax and technical skills, but it also encourages students to think critically and practice innovation. Students are required to analyze, evaluate, and creatively apply what they have learned when faced with the complex problems they are about to learn. This process greatly promotes the development of their critical thinking and innovation skills and provides a good start for students to gain a foothold in a diverse, technology-driven society in the future.

3. Instructional Design Based on UbD Theory - An Example of Selection Structure Sequence in C Language

According to the three phases in the UbD theory, a detailed instructional design is carried out as an example of a choice structure program in Chapter 4 of C Programming.

Stage 1: Determine the expected results

Based on the content, three-dimensional teaching objectives were developed as shown in Table 1.

Table 1. Educational Objective

Target dimensions	Concrete objective
Knowledge & Skills	1. Students will be able to understand and master the basic syntactic structure and usage of if-else statements and switch-case statements in C language. 2. Students can recognize and solve logical problems involving conditional judgments, such as the application of comparison operators and logical operators. Students can abstract real-world problems into logical judgments and implement them in C programs by selecting structures. 3. Students can accurately distinguish and apply different types of choice structures (e.g., single choice, multiple choice) to solve real-world problems. Students can design and select appropriate choice structures to implement program logic based on problem requirements. 4. Students can write simple C programs that incorporate choice structures and understand the importance of choice structures in program design. Students will be able to debug and fix errors in C programs that contain choice structures. Students will be able to write and run complex choice structure programs with nested if-else or switch-case statements.
Process & Steps	1. Experience the complete process from understanding the problem, analyzing the problem, and designing the choice structure to writing and debugging the C program. Experience the cyclic iterative process of writing, testing, and optimizing code in practice to improve programming skills. 2. Master the use of tools such as flowcharts and pseudo-code to assist in the design of program logic. Learn to use debugging tools (e.g., breakpoints, observing variable values) to locate and solve problems in programs. 3. Develop an approach to programming thinking that breaks down large problems into smaller ones and solves them step-by-step. Encourage creative thinking by trying different combinations of choice structures to solve the same problem and comparing their advantages and disadvantages.
Emotional Attitude & Values	1. To stimulate students' interest and enthusiasm in C programming, and to cultivate the spirit of continuous learning and exploration. To enhance communication and collaboration among students through teamwork to complete projects. To develop students' attitude of perseverance and exploration when facing programming challenges. 2. Cultivate students' scientific rigor and focus on code readability, maintainability, and efficiency. Encourage students to face failures in programming positively, learn from mistakes, and keep improving. Guide students to establish correct values and realize that programming is not only a technical activity but also a problem-solving tool that serves society and life. 3. Emphasize programming ethics and morals, such as respecting intellectual property rights and abiding by programming norms. Cultivate students' innovative spirit and practical ability, and encourage them to apply what they have learned to solve practical problems. Enhance students' logical thinking and problem-solving skills through programming practice, laying a solid foundation for their future careers.

Stage 2: Determining suitable assessment evidence

According to the content and teaching objectives of the C Language Selection Structure course, teachers can also categorize the types of assessment into three main categories, which are theoretical knowledge mastery, practical ability assessment, learning attitude, and teamwork ability assessment from the following aspects, as shown in Table 2.

Stage 3: Designing learning experiences and teaching activities

According to the course objectives and diversified assessment methods, the teaching activities are mainly divided into two parts: theoretical teaching and practical teaching.

First of all, for the teaching of theoretical foundation, the teacher's teaching activities are mainly based on lectures, and select typical cases of C language programs containing selection structures to guide students to analyze the program structure, understand the algorithmic ideas, and discuss their strengths and weaknesses. At the same time, questions are interspersed in the explanation process to check students'

understanding of the knowledge points and answer their doubts in time. Utilizing the group teaching method, students are allowed to discuss the application scenarios of selecting structures in real problems in groups and try to give solutions.

Table 2. Evaluation of evidence

Evaluation dimensions	Evaluation measures
Theoretical knowledge Mastery	1. Students will be able to accurately articulate the definition of choice structures, including single-branching (if statements), double-branching (if-else statements), and multiple-branching (switch statements). 2. Students will be able to correctly write if statements, including the writing of conditional expressions, the definition of statement blocks (using curly brackets), and the handling of nested if statements. 3. Students will be able to correctly write if-else statements, including conditional expressions, block definitions of two branches, and handling of nested if-else statements. 4. Students will be able to correctly write switch statements, including expression evaluation, matching of case tags, use of break statements, and handling of default clauses. GE.
Practical skills assessment	The student can write a program based on a given need or problem, choosing the appropriate choice structure (if statement, if-else statement, switch statement) and run it successfully to get the expected result. Students can independently identify and correct syntax errors, logical errors, etc. when writing a choice structure program to ensure that the program will run correctly.
Learning Attitude & Teamwork	Comprehensive assessment through classroom performance observation, homework submission, laboratory project participation, and communication and interaction with the instructor and classmates. Students' motivation, initiative, and conscientiousness in classroom learning, homework completion, and laboratory projects. Assessments were made through team project reports, mutual evaluation of team members, and teacher observation. In programming projects or experiments that require teamwork, students can communicate effectively and collaborate with team members to accomplish tasks.

As a subject that requires practice, C language needs to set up a practical session of writing code after completing the theoretical teaching. Teachers can assign programming assignments in class, requiring students to write C programs containing if-else statements and switch-case statements to solve the problems given in the questions. The difficulty of the assignments should follow the criteria of easy to difficult and gradually increasing, from simple conditional judgment gradually increasing to complex nested choice structure. After completing the assignments, code review activities are organized in the form of group checking, where students check each other's completion of the programming assignments, identify and correct errors, and provide guidance and advice during the session to ensure the teaching and learning objectives.

After completing the whole class, the teacher can assign students to design experimental projects after the class, such as “student performance management system”, “weather query system”, etc. Students are required to complete the project in teams, which must include the application of a selection structure. After the completion of the experimental project, students are required to write a project report, summarizing the problems encountered during the project, solutions, as well as gains and experiences. Students' logical thinking ability, problem-solving ability, and expression ability are assessed through the report. For more capable students, the instructor can also require students to think and solve problems independently by setting a series of programming challenges related to choice structures.

4. Conclusion

The main content of this unit of instruction is C Selective Logic Structures Programming, which was practically taught in the classroom after completing the instructional design through the UbD Theory Model. Reviewing the expected learning objectives after the teaching, it was found that most students performed well in understanding the basic concepts of selective structures, but there were still some challenges in applying them to practical programming, especially in dealing with complex conditions and logic. This suggests that teachers need to strengthen the theoretical foundation and design more leveled practical tasks to enhance students' application ability in future teaching.

In the teaching activities, the use of example analysis to introduce theoretical knowledge and the use of group discussion have stimulated students' interest in learning and improved classroom participation, but the implementation of the “Learning by Doing” strategy has also revealed the lack of sufficient guidance and support for some students. Therefore, teachers should strengthen personalized feedback and provide more detailed guidance and assistance to students in future teaching.

References

[1] Chen Yan. (2024). High school information technology unit learning design based on UbD theory. Pudong Education(06), 49-54.

[2] Zhou, Lan. (2023). UbD Reverse Instructional Design for C Programming Course. Electronic Technology (02), 304-306.

[3] Yu, Xiaoping, Li, Zhigang, Clan Building & Chen, Min. (2023). Java Programming Teaching Practice Based on UbD Concept and Constructivism Theory. Computer Education (02), 54-57+63. doi:10.16512/j.cnki.jsjjy.2023.02.015.