Reform of Computational-thinking Based Teaching Patterns for Programming Course

Juanqin Wang, Jianmin Sun, Yong Chen, Pan Pu

College of Information Engineering Northwest A & F University, Yangling, Shaanxi 712100, China

Abstract

Focused on the nurturing of computational thinking, this study puts forward the idea to reform the in-class teaching model, to renovate the teaching cases, and to employ diversified assessment system. It stressed that under the prerequisite of fully understanding of the curriculum objectives, teachers should let the students learn to solve problems by computer. In the process of teaching, classic cases should be employed, meanwhile, the comprehensive cases which closely relevant to what the students are learning should also be complemented, thus let the students learn the computational thinking systematically. In addition, diversified assessment system should be established to ensure the curriculum objectives would be achieved and teaching requirements would be followed.

Keywords

Diversified assessment, computation thinking, educational reform, Programming foundation.

1. Introduction

Computational thinking is a hotspot of educational transformation research of computer courses in colleges and universities both at home and abroad. In March 2006, Professor Jeannette M. Wing, Department Head of Computer Science in Carnegie Mellon University in the US, proposed the definition of computational thinking: computational thinking is to utilize the fundamental concepts of computer science to perform problem solving, systemic design and human behavior comprehension and a series of conceptual activities that cover the breadth of computer science [1]. This definition adds to more advanced and scientific connotation to the teaching ideas, orientation and requirements of computer science education. In July 2010, the first “C9 Computer Basic Course Seminar” was held by Xi’an Jiaotong University. During the seminar, a series of consensus was reached via discussion, whose core was: have a correct understanding of the significant status of computer basic teaching in colleges and universities and take the cultivation of students’ “computational thinking” ability as the core task of computer basic teaching so as to form a more complete computer basic course system and teaching contents [2].

The “C9 Computer Basic Course Seminar” took the cultivation of students’ computational thinking ability as the core task of computer basic teaching, which does not mean that the existent computer basic teaching course system and teaching contents will be “ripped and replaced” [3]. Instead, the teaching contents even the course system will be re-organized and arranged in terms of the cultivation of the computational thinking ability so as to highlight the cultivation of the thinking ability.

2. Foundation of Programming Design Teaching and Computational Thinking Ability Cultivation

"Foundation of programming design" is an important part of the computer basic teaching system in colleges and universities as well as a typical course which implements training of computational thinking method. It involves a large number of algorithms such as enumeration, recursion, iteration and sorting that are undoubtedly typical cases of computational thinking. The process and object of programming design also lie in computational thinking. Although computational thinking is more than programming design, but the final realization of computational thinking requires programming design, which is the key point of computational thinking teaching [4]. The objectives of teaching transformation of foundation of programming design are to cultivate students' computational thinking ability in the programming design course and equip every student with the basic scientific thinking ability of "computational thinking", just like demonstration thinking and logical thinking.

Although the programming design language is close to mathematics and life that are familiar to people to the greatest extent, programming design still contains strange thinking mode that students are not familiar with, forming the unavoidable difficulties in teaching and learning in the process of teaching. Inappropriate teaching process not only makes students feel far from computers but also leads to their misunderstanding of computers, let alone their better application of computers to solve professional problems. Thus, transformation of foundation of programming design course teaching is faced with remarkable challenges [3].

3. Implementation of Computational-Thinking-Oriented Teaching Schemes and Teaching Activities

3.1 Defining Teaching Objects of Foundation of Programming Design Course

For many non-computer-major students who will not engage in programming, they are not aware of the close relation between programming design and their majors; they think they don't have so many opportunities to program in the future and they don't know why they should learn this course, thus making them lack the initiative of learning and the activity of exploring and innovating. Therefore, it is of remarkable significance to make students realize the objectives and importance of learning the programming design course via foundation of programming design teaching, which will conduce to improving their interest in learning and their learning initiative.

It is just the same with the students learning mathematics and physics courses to cultivate their quality and ability of scientific thinking through training rather than to be mathematicians or physicists. The objective of learning the programming design course is to make them get hold of some foundational knowledge, techniques and methods of computer and abilities to solve problems in their own professions with the computer so that they can consciously imitate and introduce the philosophies, techniques and methods in computer science to their own professions… and utilize computers and recognize and tackle with possible problems that the could meet with in computer application. In short, the objective of learning the foundation of programming design course is to train students to use techniques and methods of computer science to handle problems and cultivate their computational thinking.

Although computing itself is a discipline, it is conducive to development of other disciplines [5]. The most essential frontier research of science and the most promising frontier research of economics of the 21st century are possible to perform with the advanced computing technology and science. Thus, it is of remarkable significance to cultivate students' computational thinking so as to enable them to research and solve problems of their own professions and make innovation.

3.2 Cultivating Computational Thinking via Diversified Teaching Methods

Content structure of the foundation of programming design course: including grammar and algorithm, wherein grammar only requires knowing and memorizing it and algorithm is the key and difficult point of this course. The course requires students to solve practical problems with the computer, which belongs to the content of computational thinking [4] and is the key point of the cultivation of computational thinking ability.

In-class teaching is the foot-stone of the whole teaching activities. There are two points in the in-class teaching of the foundation of programming design: first, through in-class teaching, students should "learn" methods and techniques to solve problems via computers. Heuristic teaching can be adopted. In this period of teaching, special attention should be paid to good interaction between teachers and students so as to arouse the learning interest of students. For instance, after explaining an algorithm, a proposition should be given. The proposition can be relative algorithm problems or application problems of this algorithm. Students can discuss it in group, design the algorithm, and present their results. Teachers will analyze and comment on algorithms and implementation methods of each group so as to make the students realize that there are various solutions to one problem and understand the method of evaluating efficiency of algorithm. Through this interactive, vibrant, discussion-oriented, opinion-exchange and exploration based class, the teaching effect of the foundation of computational thinking-oriented programming design course is guaranteed, and the students' enthusiasm in learning is inspired, which is of great importance for the cultivation of computational thinking.

Meanwhile, comprehensive example programs that correspond to the knowledge level of students are provided so as to improve and sublimate the knowledge. Good examples can narrow the distance between students and practical development environment and make students apply what they have learned. Not only can the students' interest in learning be aroused, but they can also have a more profound understanding of the relative programming design principles. For example, comprehensive subjects are provided in each chapter. After learning the knowledge of "Fundamental Structure of Programming Design" Chapter, comprehensive example programs of "gene information processing" are provided: there are welcome interface, simple menus and the achieved functions: (1) processing the DNA sequence: calculating the length of the sequence; proportion of the basic groups of A, T, C, G; and outputting another single chain that corresponds to the sequence. (2) Processing the RNA sequence: calculating the length of the sequence; proportion of the basic groups of A, U, C, G; and outputting the DNA sequence that transcribes the sequence. And (3) processing the undetermined sequence: judging the type of the sequence (DNA/RNA/ UNDETERMINED) (If U accounts for 0%, then it is DNA; if T accounts for 0%, it is RNA; if both U and T account for 0%, the it is undetermined).

Comprehensive example programs of "Luck Draw Carnival" or "Roll Calling Program" are provided after the "Array" Chapter. There is a welcome interface of the program. Students who participate in either of the programs will have their names repeatedly rolling on the screen. After pushing any of the buttons, a student is selected. If you want to continue, then operate according to the instruction and continue with the above process. You can also choose to quit from drawing or calling the roll. And all of the students selected will have their names displayed on the screen.

Example program of "Parenthesis Matching Issue" is provided after the "Pointer" Chapter. The function is that: input a series of character string consisting of parentheses from the keyboard to judge whether the parentheses match with each other. For instance, ([ ] ( )), [([ ] [ ])] , and [( ]). If the parentheses match, then output "parenthesis match". Otherwise, the unmatched type will be output: the left and right parentheses are different; the left parentheses are redundant; or the right parentheses are redundant.

Example program of "Dicing Game" is provided after the "Function" Chapter. The function is that: simulate a dicing game (two dices). The first round, if the count is 7 or 11, the player wins; if the count is 2, 3 or 12, the player loses; otherwise, the count is called "objective" and the player continues to play.

Then, if the count is “objective”, the player wins. And if the count is 7, the player loses. Other conditions are ignored and the player continues to play the game. In every round when the game is finished, the program will inquire the user if he or she wants a second try. If the user inputs words rather than y or Y, then the program will display times of failure and the game is over.

Example program of “Maintaining Database of Address Book” is provided after the “Structure Union and Union”; and example program of “Typing Exercise System” is provided after the “Document” Chapter. These comprehensive program cases can improve students’ learning interest, make them understand the programs with what they learn, know about the comprehensive programs, learn about the systemic programs whose scales are gradually increasing, and experience and comprehend the basic methods and thinking mode of utilizing computers to solve problems.

The programming design courses are of strong practicalness. If we want to solve an issue via a computer, we have to practically operate the computers from issue analysis, algorithm designing and programming. We should strengthen our comprehension, establish supporting exercises and experiment project libraries of in-class teaching and arrange the knowledge points of each chapter in graded distribution according to requirements in the teaching syllabus so that students can find problems, come up with problems and analyze problems through exercises, get hold of knowledge points of each chapter, and deeply understand the practical application methods and techniques of the knowledge points.

3.3 Diversified Assessment Modes are Conducive to Cultivation of Computational Thinking Ability

Diversified assessment modes of courses can be adopted and the final grades consist of in-class grades and assessment grades. In-class grades contain in-class quiz, homework, attendance and Q&A. In-class quiz is arranged according to the teaching week. Every week before the class, students will have a quiz on the knowledge points and classical algorithms that they learn last week. The quiz is aimed to encourage students to timely review, absorb, and apply the knowledge after the class so as to lay a solid foundation to further learning and avoid the phenomenon of cramming before the examination. The programming homework of each chapter will be arranged. There is relatively more basic knowledge that requires memorizing and learning in the first three chapters. Paper homework can be arranged. Students are required to program on the paper and then debug the program when they use the computers. If there are any problems, they should correct and debug their programs, write the correct answers on the paper homework, and take notes so as to make a reference when they review the knowledge, by which students’ good habit of programming is cultivated. In the following chapters, the programming homework can resort to “online automatic judging system”. The exercises are offered and corrected online to cultivate students’ rigorous logic thinking and designing and debugging abilities. The assessment grades have two forms and students can choose either of them. The first is to participate in the final computer examination; the second is to apply for exemption from the examination. But the students should meet the following requirements: (1) earnestly take part in the complete teaching activities (in-class listening, in-class quiz, and homework); (2) hand in a systemic program relative to the profession that the student accomplish on his or her own (or systemic programs that the students are interested in (200 lines or so)) and a complete file on this document (the document template is provided by teachers) and make a presentation in a small range. Within this process, students can come to the teachers for guidance, and the teachers will give grades according to their performance. This assessment method that utilizes the middle-sized systemic program of design development can promote the training of computational thinking and propel the students to proactively think over, discover the opportunity to use the computer knowledge and techniques to tackle with practical problems they face in their profession or life, and improve their activeness and initiative of learning, applying knowledge and innovating; (3) complete notes of the whole course (hand written, including preview notes, in-class notes and internship notes); and (4) hand in a summary (evaluate this assessment method). Students who apply for exemption from the examination can also participate in the final exam and the higher grade will be recorded.

The diversified assessment modes in our university are accepted and highly praised by non-computer-profession students of science and engineering in our university. The in-class assessment can help students to correctly understand and evaluate their knowledge and timely adjust the learning methods and schedule; and the middle-sized systemic program of design development needs fundamental knowledge and abilities of respectively comprehensive programming design and discovery, exploration and thinking in learning and lives. For students, to learn with demands is conducive to improving their activeness and initiative of learning, their motility of self-learning, and promoting research-style learning and expansion and innovation abilities.

4. Conclusion

The teaching method is elaborately designed; the classical and comprehensive cases are taken as the basis; the diversified assessment serves as the method; and course teaching of foundation of programming design and cultivation of computational thinking are combined so as to make students learn the principles, methods and techniques of solving problems via computers, initially cultivate their awareness of solving problems via computers, realize the advantages and limitation of computers to people, and finally consciously apply computational thinking to viewing, thinking over and solving problems, thus reaching the goal of cultivating and training the quality and ability of computational thinking.

Acknowledgements

This paper was financially supported by Teaching Reform Project of Northwest Agriculture and Forestry University in 2017 (No.JY1703083) and Teaching Reform Project of Shaanxi Higher Education(No.15BZ12)

References

[1] Jeannette M. Wing. “Computational Thinking”. Communications of the ACM, 2006, 49(3), p33-35.

[2] “The Joint Statement of the Development Strategy for Basic Computer Teaching of Nine University Alliance (C9)”.China University Teaching, 2010(9),p4-9.

[3] Qinming He, Hanquan Lu, Boqin Feng. “The Core Task of Computer Basic Teaching is to Cultivate the Ability of Computational Thinking”. China University Teaching, 2010(9), p5-9.

[4] Peizeng Gong, Zhiqiang Yang. “Computational Thinking Training in the Computer Basic Teaching at University”. China University Teaching, 2012(5), p51-54.

[5] President’s Information Technology Advisory Committee. Computational Science: Ensuring America’s Competitiveness [EB/OL]. http://www.nitrd.gov /pitac /reports /20050609_computational/computational.pdf, June 2005.

[6] Guoliang Chen, Rongsheng Dong. “Computational Thinking and Computer Basic Teaching of University”. China University Teaching, 2011(1), p7-10.