Improving Learning Efficacy in Basic Programming Practice based on Three-stage Thinking

Xiaojun Cai
School of Computer Science and Technology
Shandong University
Qingdao, China
xj_cai@sdu.edu.cn

Jacky Keung
Department of Computer Science
City University of Hong Kong
Hong Kong, China
Jacky.Keung@cityu.edu.hk

Zhaoyan Shen
School of Computer Science and Technology
Shandong University
Qingdao, China
shenzhaoyan@sdu.edu.cn

Mengying Zhao
School of Computer Science and Technology
Shandong University
Qingdao, China
zhaomengying@email.sdu.edu.cn

Zhen Yang
School of Computer Science and Technology
Shandong University
Qingdao, China
zhenyang@sdu.edu.cn
(Corresponding Author)

Zhenge Jia
School of Computer Science and Technology
Shandong University
Qingdao, China
zhengejia@email.sdu.edu.cn

Fangxi Han
School of Computer Science and Technology
Shandong University
Qingdao, China
hfx@sdu.edu.cn

Dongxiao Yu
School of Computer Science and Technology
Shandong University
Qingdao, China
dxyu@sdu.edu.cn

***Abstract*—Basic Programming Practice (BPP), as an introductory course for computer science majors, aims to enable students to have basic programming skills and lay a foundation for subsequent advanced course learning. This article draws on the concept of "using qi and cultivating both inside and outside" in traditional Chinese martial arts and highlights the importance of "thinking style" as an internal factor, developing a Three-stage Thinking (TT) methodology to foster students' learning efficacy in programming practice. In detail, TT consists of (1) orderly and logical thinking, (2) mathematical and computational thinking, as well as (3) linear and systematic thinking. Accordingly, we design specific and novel teaching methods to guide students to conduct the thinking practice following TT in programming. Extensive teaching experience has proved that the TT has effectively improved students' programming entry efficiency and design ability. Follow-up survey data tracking over half a decade shows that this method strongly supports students in programming learning. Thus, we highly recommend using TT methodology in BPP-related courses to unleash students' potential.**

Keywords—Basic Programming Practice, Three-stage Thinking, Teaching Experience, Chinese Martial Art

I. INTRODUCTION

The Hollywood animated series "Kung Fu Panda" has swept the world several times, showing the charm of Chinese Kung Fu to global audiences [11]. The scene in the movie where the dragon warrior Po practices "qi" truly reflects a way of practicing traditional Chinese martial arts, and "qi" is also the most mysterious and respected part of traditional Chinese martial arts. Traditional Chinese Kung Fu has always emphasized: "training one's breath internally and practicing one's muscles, bones and skin externally", where "muscles", "bones", and "skin" refer to the external form and moves, while "qi" refers to the internal cultivation. Taking "qi" first is the key to truly reflecting the advancement of martial arts cultivation. In particular, the "Tai Chi" school in Chinese martial arts believes that it is more about intention than form. When practicing, do not only focus on the form and moves of the standard boxing frame. The form is only the expression of the inner intention, and the intention is the embodiment of qi. Without intention, the form is just a facade. The practice of Chinese martial arts can be simply divided into three stages according to the mastery of "qi": entry, mastery, and perfection. The entry stage refers to a preliminary understanding of "qi", while mastery, as a small achievement, refers to a certain grasp and application of "qi". When one has mastered "qi" proficiently and applied it flexibly, one can achieve "free movement", which can be regarded as the perfection of a martial art.

In the design of training programs for computer majors in many universities, BPP is offered as the first professional basic course, shouldering the mission of enlightenment, foundation building, and ability training. Many students, and even teachers, simply think that after mastering the syntax of programming languages, the rest is to practice and do exercises continuously[8],[9]. Nevertheless, in practice, we found that this learning process does not solve the students' inherent way of thinking, and often leads to slow entry, slow improvement, and poor results. When students encounter practical problems, they always sigh: Why can't I think of it?

Based on our long-term teaching experience, we argue that students' study of this course is like practicing a Chinese martial art. To cultivate the inner "qi" well, students should first pay attention to the understanding and grasp of the inner way of thinking, gradually acquire the way of thinking to analyze and solve case problems, know where to start, have a thinking process, and propose reasonable methods. Through the teaching practice of C language programming, this article summarizes the three-stage thinking method to facilitate basic programming learning at the university level, as shown in Figure 1. Specifically, (1) Through case analysis on the definition and connotation of orderly thinking, helping students understand the way of logical thinking, gradually establishing a logical and orderly thinking consciousness. As such, realizing the understanding and analysis of the program process. (2) Through case comparison between mathematical solutions and programming solutions, analyzing the similarities and differences between mathematical thinking and computational thinking, thereby, helping students to achieve a unity of mathematical thinking and computational thinking. (3) Through case design, helping students to change from a linear thinking mode that focuses on the linear causality of problems to a systematic thinking mode that explores the diversity and complexity of problems, thereby

2025 International Symposium on Educational Technology (ISET) | 979-8-3315-9550-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISET65607.2025.00048

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 03:20:38 UTC from IEEE Xplore. Restrictions apply.

improving their ability to solve complex engineering problems. According to the above methodology, we conducted extensive practice and tracking surveys during the last six years and testified to the effectiveness of the TT methodology for training students' programming thinking and skills.

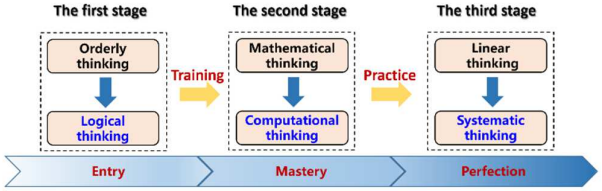

Fig. 1. Three-stage Thinking Methodology.

II. THREE-STAGE THINKING

The learning process is a gradual process. Through the three-stage thinking methodology, students can establish a good way of thinking at the beginning of their learning, instead of being confused by grammar rules and example learning during their programming. The establishment of a way of thinking can allow students to know the truth and the reason, achieving twice the result with half the effort.

A. The first stage thinking

Orderly thinking means that things should be done in order and steps, which is also the inherent way of thinking and the work process of human beings. After understanding the basic grammatical rules and program structure of C language, students often focus on the two structures of conditional and loop, and ignore the fact that program structure is first of all a sequential structure. When programming, students do not fully consider what to do first and what to do later, but quickly fall into the trap of branches and loops. That is why error information such as *[Error] "XXXX" was not declared in this scope* always appears during their debugging period. In teaching practice, we try to let students view the conditional structure and loop structure as a whole, or even as a compound statement, and the entire program will eventually present a sequential structure. This orderly way of thinking is conducive to beginners' mastery of the program design process and understanding of program modular design.

Programming is orderly, and the rules and reasons for order are what logical thinking needs to solve. Logical thinking can be seen as an explanation and extension of orderly thinking. It is a methodical, well-founded, and highly analytical way of thinking. In programming, the appearance of each statement must meet all its prerequisites. For example, if there is a statement *sum=sum+3;* in the program, the definition and initialization of sum must appear before the statement. It is not difficult to see that logical thinking strengthens the logic of orderly thinking. In the teaching practice of a class with 150 students (in one semester), when students were asked to answer the number of times the loop in the program segment *n=0; while(n++<10);* was executed, all students gave the correct answer of 10; but when asked about the value of n after the program segment was executed, only a few students gave the correct answer of 11. This is because students can understand that *n++<10* is the sequential execution of two expressions, i.e., self-increment (*n++*) and conditional expressions (*n++<10*). Thus, they know the execution order of the while loop, and can quickly calculate the number of loops. But students do not understand the logical relationship between *n<10* and *n=n+1*, so they do not know that when *n<10* is not true, *n=n+1* still needs to be executed one more time.

To further help students understand and master the two ways of thinking above, we asked students to design solutions to the n-th term of the Fibonacci sequence [10] and counted the methods used by the students. The results are shown in Table I. The results show that all students summarized the rules of the sequence and completed the program design, reflecting basic logical thinking ability. Among them, 58% of the students used array methods to achieve it, demonstrating that orderly thinking dominates most students' thinking mode. 42% of the students obviously had more experience in logical thinking, especially the last solution in the table, accounting for 14% of the students, indicating that they had a deeper grasp of logical thinking.

TABLE I. STATISTICS OF PROGRAMMING METHODS ON FIBONACCI SEQUENCE.

Method	Number	Ratio
int a[N]; a[n]=a[n-1]+a[n-2]	87	58%
t=f1+f2; f1=t-f1; f2=t;	42	28%
f2=f1+f2; f1=f2-f1	21	14%

B. The second stage thinking

Students have a long history of learning mathematics and have a strong inertia to solve problems using mathematical methods, which often restricts their understanding and grasp of computational thinking [1]. As shown in Table II, when solving "Decompose 3-digit positive integers", all students chose Method-1. It is not difficult to see that Method-1 is a manifestation of logical reasoning. In essence, it uses mathematical thinking to solve problems, which is more like manual implementation than using computers to solve problems. Subsequently, we changed the topic to "Decompose *n*-digits positive integers". Since the value of *n* was uncertain, it was difficult to solve it using mathematical thinking. As a result, 118 students did not give a good design within the required time limit. 32 students completed the design using Method-2 which fully embodies the idea of computational thinking and truly allows computers to solve problems.

TABLE II. STATISTICS OF PROGRAMMING METHODS ON POSITIVE INTEGERS DECOMPOSITION.

Subject	Method-1	Method-2	Unrealized
	a=n/100; b=(n-a*100)/10; c=n a*100b*10;	***n%10; n=n/10;***	***N.A.***
Decompose 3-digit positive integers	150	42	0
Decompose n-digit positive integers	0	21	118

Computational thinking needs the support of mathematical models and reasoning processes. The integration of the two ways of thinking is what students really need to understand. In teaching practice, we borrowed the mathematical problem raised by the ancient Chinese mathematician Qiujian Zhang in his book *Suan Jing* [14] and constructed a programming problem for students: "One rooster is worth five coins, one hen is worth three coins, and three chicks are worth one coin. If you buy 100 chickens with one coin, how many roosters, hens, and chicks are there?", namely the Hundred Yuan Hundred Chickens (HYHC) problem, thereby further

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 03:20:38 UTC from IEEE Xplore. Restrictions apply.

explaining to students the relationship between mathematical thinking and computational thinking.

To reduce the complexity of this problem, we first asked students to implement the program design of "buying the least chickens (with a full range of varieties) with 100 yuan", namely the Hundred Yuan Minimum Chickens (HYMC) problem. As can be seen from Table III, only 29 students used the enumeration method to solve the problem, and these 29 students also used the same mathematical thinking method as other students to complete the program design. After all, mathematical reasoning is more convenient and easier.

TABLE III. STATISTICS OF PROGRAMMING METHODS ON HYMC AND HYHC.

Subject	Mathematical meth. (equations)	Computational meth. (enumeration)	Unrealized
HYMC	121+29	29	0
HYHC	0	99	44
	7		

As for the HYHC problem, 99 students understood the design method of the enumeration algorithm very well and deepened their understanding of computational thinking. However, 44 students did not give a good design and still needed further experience, which is inevitable. When martial arts reaches a certain level, it needs a longer period of consolidation and accumulation. However, 7 students in Table III did not stick to one way of thinking. They first derived the following equations through mathematical derivation:

$$\begin{cases} x \le 14 & (1.1) \\ 7x = 4(25 - y) & (1.2) \end{cases} \quad (1)$$

Here x is the number of roosters, y is the number of hens, and according to equation (1.2), it can be seen that x must be divisible by 4. The seven students designed programs based on this mathematical deduction, optimizing the enumeration design through three loops into a program with one loop. This is a model of integrating mathematical thinking and computational thinking to solve problems, achieving the best results at this stage.

C. The third stage thinking

Linear thinking is a linear, unidirectional, and single dimensional way of thinking. When dealing with problems, it tends to simplify complex situations into linear cause and-effect relationships, ignores the multiple, complex, and dynamic connections between things, and pays more attention to local factors and results. Systematic thinking is a transcendence of traditional thinking. It can be understood as a system view, emphasizing the comprehensiveness, integrity, and dynamism of looking at problems, and focusing on the interrelationships between problems and the combined impact of various factors.

In teaching practice, we analyze the classic knapsack problem [7] and let students compare the characteristics of the two ways of thinking, by which students understand that the enumeration method can get the optimal solution to the problem, but the calculation scale is large. At the same time, using linear thinking and greedy algorithms to quickly get an approximate solution. As the complexity of the problem increases, we should not only learn how to simplify and use linear thinking to quickly obtain results but also have a systematic understanding of the overall problem and comprehensively consider the impact of various factors. In the homework left to students, we deployed a series of questions as shown in Table IV. Through this step-by-step homework design, students are encouraged to have the ability to simplify complexities, while strengthening their multi-dimensional and multi-level systematic thinking literacy. Through such training, students show good comprehensive literacy in the later course project stage.

TABLE IV. EXAMPLE OF HOMEWORK

Large number addition: Enter two large numbers and calculate their sum.
Evaluation method and data scale:
– For 12% of the data, the two numbers entered are positive integers.
– For an additional 24% of the data, the two numbers entered have no decimal point.
– For an additional 32% of the data, the two numbers entered have a fixed number of 2 decimal places after the decimal point.
– For 100% of the data, the existence of the decimal point is uncertain, and if it exists, the number of decimal places after it is also uncertain.
– - The length of the input number does not exceed 1000 digits.

III. ACHIEVEMENTS

We adopted our proposed TT methodology in the BPP course and investigated its effectiveness in three semesters (from fall 2022 to fall 2024) via a comprehensive survey, where 443 out of 450 feedbacks were received from year-1 undergraduate students who attended the BPP course at our university. Each questionnaire consists of three questions, including (1) "To what extent can TT methodology facilitate the program understanding.", (2) "To what extent can TT methodology contribute to the mastering of programming skills.", and (3) "To what extent can TT methodology contribute to planning complex problems." Particularly, we employ the three-level Likert scale [4] which allows interviewed students to express their degrees of positive or negative attitudes based on their experience and observations, from "High", "Medium", to "Low" for results collection. As shown in Figure 2, Results revealed that 90.07% of interviewees consider TT methodology highly facilitate their understanding of programs, 94.13% of them emphasize the high importance of TT methodology in improving programming skills, and 98.65% of them acknowledge that TT methodology highly contributes to their coding planning when solving complex problems. In particular, all interviewees consider TT methodology to have at least medium positive effects on their programming skills and ability to solve complex problems.

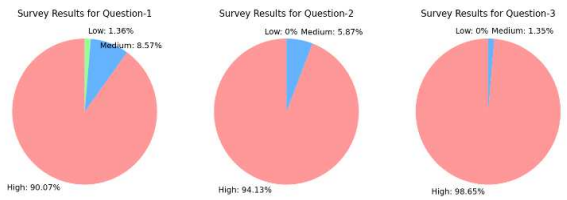

Fig. 2. Survey results for three questions

Students' understanding and grasp of the three-stage thinking effectively improved their programming ability,

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 03:20:38 UTC from IEEE Xplore. Restrictions apply.

eventually contributing to their performance in developing software projects. During the last six years, we tracked 900 year-1 undergraduate students from fall 2019 to fall 2024 and assessed their performance in developing the Gomoku Game after the one-semester learning of BPP, where students in the first three years were not taught with TT methodology while the latter three years were. In detail, we divided the implementation of the Gomoku Game into three stages, namely (1) basic Human-to-Human battle (H2H) function, (2) Human-to-Machine battle (H2M) function, and (3) advanced functions, e.g., user interface, game temporary storage, and replay, corresponding to the level of three-stage thinking, i.e., entry stage, mastery stage, and perfection stage. Table V shows the detailed statistics before and after using TT methodology in BPP course teaching. As can be seen, before teaching with TT methodology, all students have achieved the basic human-to-human battle function, of which 80 students can complete the human-to-machine battle function, and only 23 students can implement advanced functions, such as realizing the user interface, game temporary storage, replay, and other systematic modules. After teaching with TT methodology, students' performances in this course project development are boosted significantly, where the number of students who only complete entry-level functions decreases to 170, while students who can accomplish the mastery-level and perfection level functions increase to 146 and 134, demonstrating a great success in the application of TT methodology.

TABLE V. STATISTICS OF DEVELOPING PERFORMANCE IN THE GOMOKU GAME.

Function / Degree	Human-to-Human	Human-to-Machine	Advanced
Entry Level	347/170	N.A.	N.A.
Mastery Level	80/146		N.A.
Perfection Level	23/134		

Every academic year, we conduct follow-up surveys on previous students, and the feedback data shows that our TT methodology has a good support and boost effect on students subsequent professional learning and development. All of our students participate in the Certified Software Professional(CSP)[1], and the certification results consistently rank highly among the top universities in China. In addition, students have achieved good results in various subject competitions such as mathematical modeling, the China Collegiate Programming Contest (CCPC) [2], and the International Collegiate Programming Contest (ICPC)[3].

IV. RELATED WORK

In recent years, various learning methodologies have been proposed to enhance the efficacy of Computer Science (CS) education, thereby bridging the gap between conceptual understanding and practical programming skills, particularly for beginners [2], [3], [5], [6], [12]. For example, Zendler et al.[13] extensively explored 20 instructional methods within the educational domain of CS, and concluded that certain instructional methods are especially predestined for CS education, such as problem-based learning and discovery learning. Xue et al. [12] exploited the advantages of graphic programming to practice and foster elementary students' computational thinking skills. Li et al. [5] integrated generative Artificial Intelligence (AI) into software engineering education, aiming at endowing students with essential competencies tailored for contemporary software development. Nevertheless, our work provided a brand-new methodology for systematically improving undergraduate students' learning efficacy in basic programming practice and verified its effectiveness in extensive tracking and survey across over half of a decade.

V. CONCLUSION

Basic programming practice is a course aimed at beginners, helping them to develop a good way of thinking and programming skills. The extensive practice has proved that the three-stage thinking methodology proposed in this article can help students understand and grasp the connotation of orderly and logical thinking, mathematical and computational thinking, as well as linear and systematic thinking, realize the comprehensive application of the six thinking modes, effectively promote students to understand the scientific principles and scientific methods in the field of computer science and improve students' ability to analyze and solve complex engineering problems.

ACKNOWLEDGMENT

This work was supported in part by the Undergraduate Teaching Reform Research Key Project of Shandong Province under Grant Z2022153, the Class Teaching Reform Project of Shandong University, and the General Research Fund of the Research Grants Council of Hong Kong and the research funds of the City University of Hong Kong (Grant Nos. 6000871,9229109, 9229098, 9220103, 9229029).

REFERENCES

[1] Mordechai Ben-Ari. Mathematical logic for computer science. Springer Science & Business Media, 2012.

[2] Pak Yuen Patrick Chan, Jacky Keung, and Zhen Yang. Validating plagiarism detection systems with metamorphic testing. In 2023 International Symposium on Educational Technology (ISET), pages 132–137. IEEE, 2023.

[3] Michail N Giannakos, John Krogstie, and Nikos Chrisochoides. Reviewing the flipped classroom research: reflections for computer science education. In Proceedings of the computer science education research conference, pages 23–29, 2014.

[4] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. Likert scale: Explored and explained. British journal of applied science & technology, 7(4):396–403, 2015.

[5] Yishu Li, Jacky Keung, and Xiaoxue Ma. Integrating generative ai in software engineering education: Practical strategies. In 2024 International Symposium on Educational Technology (ISET), pages 4953. IEEE, 2024.

[6] Yishu Li, Jacky Keung, Xiaoxue Ma, Jingyu Zhang, Zhen Yang, and Shuo Liu. Learning gaps in project-based requirements engineering education-a case study of student projects. In 2023 International Symposium on Educational Technology (ISET), pages 239–243. IEEE, 2023.

[7] Silvano Martello and Paolo Toth. Algorithms for knapsack problems. North-Holland Mathematics Studies, 132:213–257, 1987.

[8] Yizhou Qian, Susanne Hambrusch, Aman Yadav, Sarah Gretter, and Yue Li. Teachers' perceptions of student misconceptions in introductory programming. Journal of Educational Computing Research, 58(2):364397, 2020.

[9] Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. ACM Transactions on Computing Education (TOCE), 18(1):1–24, 2017.

[1] https://www.cspro.org/
[2] https://ccpc.io/
[3] https://icpc.global/

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 03:20:38 UTC from IEEE Xplore. Restrictions apply.

[10] Tony C Scott and Pan Marketos. On the origin of the fibonacci sequence. MacTutor History of Mathematics, 23:1–46, 2014.

[11] Chenjun Wang. The western gaze in animation: A case study of kung fu panda. Journal of Content, Community & Communication, 6(3):3–12, 2017.

[12] Tianyu Xue, Shiyan He, and Weitong Guo. Design and implementation of a graphical programming class for computational thinking in elementary schools. In 2024 International Symposium on Educational Technology (ISET), pages 376–380. IEEE, 2024.

[13] Andreas Zendler and Dieter Klaudt. Instructional methods to computer science education as investigated by computer science teachers. Journal of Computer Science, 11(8):915, 2015.

[14] Luxizi Zhang. From potential to practical variations in the teaching of functions: contrasting Chinese and French cases in two upper secondary schools. PhD thesis, Ecole normale sup´erieure de lyon-ENS LYON; East China normal university (Shanghai) , 2022.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on September 17,2026 at 03:20:38 UTC from IEEE Xplore. Restrictions apply.