Enhancing Computational Thinking in Programming Learning with Generative Artificial Intelligence Tools for college students

Chengzheng Li[1], Yongfeng Diao[2] and Yi Ding[3,*]

[1]China West Normal University, Educational and information technology center, Nangchong 637000, China
[2]China West Normal University, School of education, Nangchong 637000, China
[3]China West Normal University, Admissions and employment office, Nangchong 637000, China

*ding4520@163.com

Abstract. The era of numerical intelligence has put forward higher requirements on the computational thinking of college students, and the lack of computational thinking is still common among college students at present. In an investigation to find methods and measures to cultivate college students' computational thinking that are widely applicable and effective, the study investigated the effect of artificial intelligence generated content tools in programming education to enhance college students' computational thinking through a 13-week quasi-experimental study. The results of the study show that (1) conventional programming education activities do not significantly enhance college students' computational thinking; (2) artificial intelligence generated content tools in programming education activities significantly enhance college students' computational thinking; and (3) uncontrolled use of artificial intelligence generated content tools is not conducive to the enhancement of learners' academic performance. On this basis, strategies and suggestions are proposed for better utilization of generative artificial intelligence generated content tools to enhance college students' computational thinking ability and academic performance in programming learning.

Keywords: artificial intelligence generated content, programming education, computational thinking, educational applications

1 Introduction

In the era of digital intelligence, information technology and artificial intelligence are deeply integrated into our daily lives. They are crucial for driving national progress and ensuring national security, making them vital to a nation's competitive strength[1]. Computational Thinking (CT), the fundamental thinking of computational science, refers to "a set of problem-solving activities that use the basic concepts of computer science to solve problems, design systems, and understand human behav-

H. Kannan et al. (eds.), *Proceedings of the 5th International Conference on Internet, Education and Information Technology (IEIT 2025)*, Atlantis Highlights in Social Sciences, Education and Humanities 40,
https://doi.org/10.2991/978-94-6463-803-5_13

ior"[2]. CT is considered a normal cognitive and thinking mode for the present and the future[3]. It is hailed as one of the most important abilities of the 21st century[4] and is a capability that every talent in the digital age should possess[5,6]. To this end, the cultivation of computational thinking skills has been incorporated into primary and secondary school curricula by the state. Educators and educational researchers have conducted extensive studies on this, but most of these studies focus on the K-12[7].

College students, as the main force in accelerating the development of new quality productive forces, should be cultivated in computational thinking to enhance the matching degree of talent cultivation and social needs, and attention should be paid to the improvement of learners' computational thinking abilities in all disciplines, majors, and classrooms[8]. Some scholars have already realized the importance of cultivating computational thinking in higher education and have tried to improve learners' computational thinking by combining it with computer-related professional courses, but most of the related research is still at the theoretical discussion stage, with a lack of empirical research[9]. At present, there is a general lack of computational thinking among college students and adult learners in China[10], and there is an urgent need to find a widely applicable and effective method and measures for cultivating college students' computational thinking.

2 Systematic Review of Previous Studies

2.1 Nurturing Computational Thinking

Since the advent of computational thinking, the relevant literature has mainly focused on K-12. With the deepening of computational thinking research, scholars have gradually shifted their attention to higher education. However, the related research still focuses more on the theoretical discussion, and there are few empirical studies. For example, Liu Huibin et al. carried out research on the demand for computer application ability from multiple perspectives of students, teachers, and employers, and designed a four-dimensional model for the cultivation of college students' computational thinking[11]; Li Jiaying et al. analyzed the category characteristics of college students' computational thinking through a scale test[12].

Fortunately, there are a large number of examples of practical research in K-12 that can inform research in higher education. Currently, K-12 develops learners' computational thinking in three main ways: (1) Creating a learning environment for the development of computational thinking. That refers to the creation of a learning environment that can strengthen the thinking process of problem-solving by relying on problem-solving-oriented computer science or information technology courses (e.g., artificial intelligence, information technology, programming, etc.) and designing a pathway for the cultivation of each of these elements on the basis of the basic elements of computational thinking. Shen et al. created a learning environment for solving real problems in an artificial intelligence course through the "structure-relationship-value" method[13], and Hsu et al. created a peer-interactive learning environment in a robotics education activity, which promotes the enhancement of learners' computational

thinking skills and co-operative and social skills[14]. (2) Provide learning tools that foster computational thinking. That is, providing learners with tools that reinforce the frequency and effectiveness of the application of computational thinking during learning activities. The tools commonly used by scholars are robots (programmed robots, chatbots), graphical or modular programming applications, smart toys, gamification props. For example, Lin et al. used smart toys to improve preschool children's learning behavior and enhance their interest in learning and computational thinking[15]. (3) Designing learning activities that develop computational thinking. This means carefully planning a series of educational or training activities that aim to develop and improve learners' computational thinking skills. These practical experiences in K-12, although they cannot be directly copied and applied, provide positive reference values for computational thinking development in higher education.

2.2 Programming Education Contributes to the Formation of Computational Thinking

Computational thinking has always been closely associated with programming education since its birth. Programming is widely recognized as the ideal medium for the development of computational thinking, and programming education has naturally become the main way to develop computational thinking[3, 16].

To be specific, programming provides a structured and logical approach to problem-solving, and by writing code, debugging, and optimizing programs, learners can exercise and develop computational thinking in a systematic way. Therefore, many experts have analyzed and studied the enhancement of learners' computational thinking through programming education. By analyzing 26 empirical studies, Gao Hongyu et al. found that programming education can effectively enhance the development of computational thinking in early childhood (0-8 years old), but the effect of different programming tools on the enhancement of computational thinking is not the same, in which tangible programming tools are more suitable for early childhood direct perception, hands-on experience and practical operation[17]. Zhang et al. indicated that computational thinking can be acquired through programming education by analyzing 55 empirical studies at the K-9[18]; Hsu et al. analyzed 120 articles at the K-12 over ten years (2006-2017) and found that enhancing learners' computational thinking through programming education was the main way[19]. It is evident that the role of programming education in the development and promotion of computational thinking in k-12 is well established, but the effect of programming education activities on computational thinking in higher education needs to be further investigated.

2.3 Artificial Intelligence Generated Content Into Programming Education

Despite the growing interest in learning programming, learning programming remains a challenging and complex process for most students. Many learners may choose to give up halfway through their programming studies for several reasons, including the

lack of required support and guidance during the programming process as well as the difficulty often encountered in debugging errors in the program[20].

The artificial Intelligence generated content tools provide a glimpse to crack this dilemma. Artificial intelligence generated content (AIGC) is an artificial intelligence technology that employs a deep learning approach to train large amounts of data to be able to create new content that is close to human behavior[21]. The technology has received a lot of attention in the education sector thanks to the rollout of an AIGC tool (Chat-GPT). Chat-GPT is an intelligent dialog system based on a large language model, capable of assisting learning and teaching through the input, interaction, and generation of natural language text, providing students and teachers with more convenient and efficient learning[22,23]. Similarly, the tool provides excellent support for programming learning. Specifically, the AIGC tool provides easy access, supports multiple languages, and requires no programming learning experience, which greatly reduces the threshold of programming learning; second, the AIGC tool provides fast response, allows queries and searches, and intelligent responses, which meets the personalized needs of programming learning; and lastly, the AIGC tool provides clear explanations and programming examples, as well as provides resources on high-level topics, which expands the programming learning depth and breadth[24]. However, most of the research related to the AIGC into programming learning remains in theoretical studies, and further empirical studies are necessary.

Based on this, this study aims to analyze the impact of AIGC tools on learners' computational thinking and academic performance in programming education and explore the reasons for their utility. To come up with guidance recommendations for utilizing AIGC tools to empower learners to improve their computational thinking and academic performance in programming learning. The specific research questions are as follows:

• RQ1: what is the influence of programming education activities on college students' computational thinking?

• RQ2: what changes in college students' computational thinking are produced when using AIGC tools to assist programming learning?

• RQ3: what is the academic performance of college students when using AIGC tools to assist programming learning?

3 Methodology

3.1 Participants and Experimental Settings

The subjects of the study were two randomly selected freshman classes at a university in western China, with 225 students in total, including 107 students in the experimental class and 118 students in the control class. The students in these two classes had similar basic knowledge of programming, and there was no significant difference in computational thinking ability. These students took the public course Python Programming together.

This experiment is a 13-week pre/post-test non-reciprocal control group quasi-experimental study. Specifically, the content of the course is extensive and requires a

lot of practice, so both classes adopt the blended learning mode of "online + offline". "Online" is equipped with a full set of online self-study courses, and "offline" is carried out in the form of a combination of classroom theoretical learning and practical training. The difference between the two classes is that the teacher provides each student with the AIGC tool (ERNIE Bot) as a learning assistant during the practical exercises. At the same time, a tutorial on how to use the AIGC tool and its techniques was provided. It is worth noting that only the learning performance assessment was conducted in the 13th week, and the assessment was conducted through a computerized paperless exam. In the experiment, the researcher conducted a pre-test and a post-test on the computational thinking of the students in the experimental and control classes, respectively.

3.2 Computational Thinking and Academic Achievement Testing Tools

Based on analyzing the evaluation system of computational thinking proposed by Chen Xingye and Brennan et al[25,26], and referring to the scale cases of Zhang Yi, Yang Gang, and Kang et al, a multidimensional test scale for college students' computational thinking were determined[27,28]. The measurement scale contains five ability dimensions of decomposition, abstraction, modeling, algorithm, and evaluation, with a total of 17 questions. Cronbach's alpha (Cronbacha) coefficient was used as the measure for reliability analysis. The results showed that Cronbacha=0.94 for the pre-measurement and Cronbacha=0.96 for the post-measurement, which made the reliability of the scales acceptable on the whole. Meanwhile, the results of the validity test showed that the KMO value of each dimension was greater than 0.9, and the results of Bartlett's spherical test showed Sig.<0.01, which showed good overall validity.

3.3 Data Collection and Analysis

This experiment was conducted to obtain data on both computational thinking and academic performance. Computational thinking data were collected through a line matrix; academic performance data were collected through a computerized paperless exam. Computational thinking was measured in two groups before and after the experiment. After the data were recovered, invalid data were removed if the length of the answer was too short and if the answers to all questions were the same. A total of 407 data were obtained, including 188 in the experimental group (93 in the pre-test and 95 in the post-test) and 219 in the control group (107 in the pre-test and 112 in the post-test). The collection of academic performance data took place after the experiment and 225 answer sheets were obtained, of which 107 were from the experimental group and 118 from the control group.

To analyze the impact of programming education and AIGC tools on computational thinking and academic performance, an independent samples t-test was used and the results were visualized. For the purpose of further analyzing the essential reasons for the utility of programming education and AIGC tools in learning activities, learners were randomly interviewed and exchanged after the experiments, and the ex-

changes included the methods of solving difficulties in the process of completing the project, the views on AIGC tools, the learning gains of the programming course, and the opinions and suggestions, and so on.

4 Results and Discussion

4.1 The Influence of Programming Activities on College Students' Computational Thinking

In this section, we counted the computational thinking of the control group before and after the experiment and visualized the statistics (see Fig.1). Analyzing Fig.1, we have the following observations.

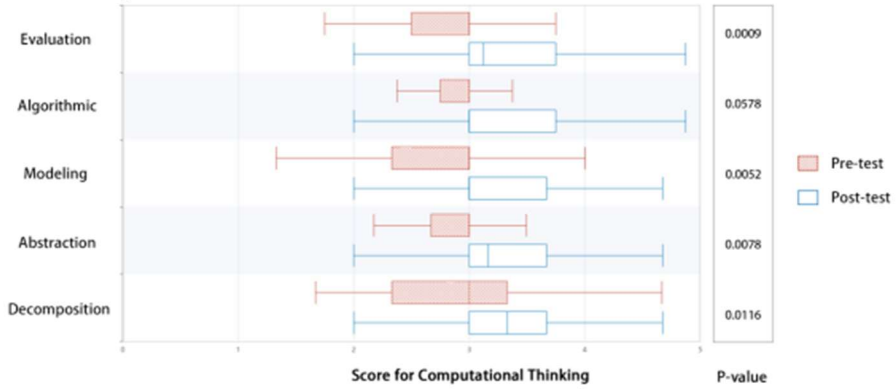

Fig. 1. Analysis of differences in dimensions of computational thinking before and after the experiment in the control group.

Overall, learners in the control group showed some improvement in computational thinking after the experiment, but the improvement was not significant. Specifically, the T-test results of decomposition ability are P=0.8116>0.05; abstraction ability is P=0.7633>0.05; modeling ability is P=0.2683>0.05; algorithmic ability is P=0.3680>0.05; and evaluation ability is P=0.5778>0.05.

Programming activities are most effective in improving modeling. In particular, learners improved by an average of 0.04 in decomposition (pre-test M=3.02, SD=0.67; post-test M=3.06, SD=0.81); by an average of 0.05 in abstraction (pre-test M=2.99, SD=0.62; post-test M=3.04, SD=0.76); and by an average of 0.18 in modeling (pre-test M=2.88, SD=0.69; posttest M=3.06, SD=0.77); in algorithmic ability by an average of 0.13 (pretest M=3.04, SD=0.61; posttest M=3.17, SD=0.69); and in evaluation ability by an average of 0.08 (pretest M=2.99, SD=0.56; posttest M=3.07, SD = 0.77).

4.2 The Impact of AIGC tools on College Students' Computational Thinking

In this section, we counted the computational thinking of the experiment group before and after the experiment and visualized the statistics (see Fig.2). Analyzing Fig.2, we have the following observations.

These results showed that the learners in the experimental group showed a massive improvement in computational thinking after the experiment, with significant improvements in decomposition ($P=0.0116<0.05$), abstraction ($P=0.0078<0.05$), modeling ($P=0.0052<0.05$), and evaluation ($P=0.0009<0.001$).

The AIGC tool improved learner decomposition, abstraction, modeling, and evaluation better than algorithms. Specifically, learners improved by an average of 0.38 in decomposition (pretest M=2.97, SD=0.68; posttest M=3.35, SD=0.61); 0.38 in abstraction (pretest M=2.93, SD=0.62; posttest M=3.30, SD=0.59); 0.46 in modeling (pretest M=2.78, SD=0.71; posttest M=3.24, SD=0.69); in algorithmic there was an average improvement of 0.26 (pre-test M=2.99, SD=0.52; posttest M=3.25, SD=0.63); and in evaluation there was an average improvement of 0.47 (pre-test M=2.83, SD=0.56; posttest M=3.30, SD = 0.66).

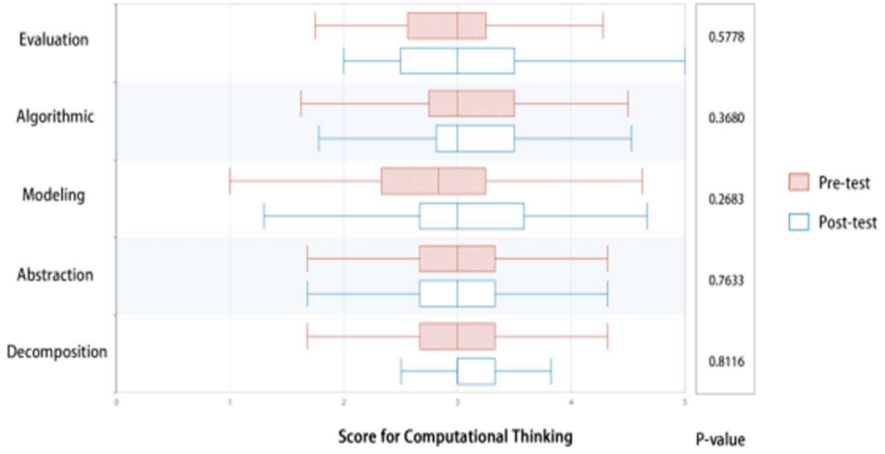

Fig. 2. Analysis of differences in dimensions of computational thinking before and after the experiment in the control group.

4.3 The Effect of AIGC tools on College Students' Academic Achievement

In this section, we counted the academic performance of the experiment group and control and visualized the statistics (see Fig.3). Analyzing Fig.3, we have the following observations. AIGC tools to promote computational thinking.

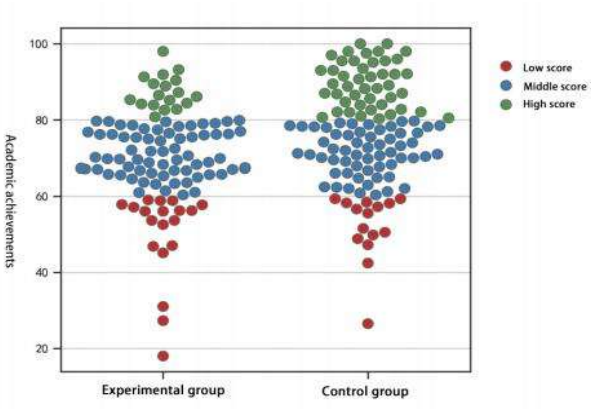

Fig. 3. Analysis of differences in academic performance between the experimental group and the control group.

The results indicate that control group (M=75.86, SD=13.89) significantly (P=0.0013<0.05) outperformed the experimental group (M=69.96, SD=13.13) in terms of academic performance, especially in terms of the percentage of low performers (academic performance<60) and high performers (academic performance>80). Specifically, the control group (N=45, or 38.14%) had more high-performing students than the experimental group (N=15, or 16.82%) as a percentage of the entire class, and the control group (N=15, or 12.71%) had fewer low-performing students than the experimental group (N=19, or 17.76%) as a percentage of the entire class.

The control group had a higher highest academic performance (Academic Performance=100.00) than the experimental group's highest academic performance (Academic Performance=98.00) and a higher lowest academic performance (Academic Performance=26.50) than the experimental group's lowest academic performance (Academic Performance=18.00).

5 Conclusion

5.1 Conventional Programming Education Activities Are Not Significant Promoting Computational Thinking Among College Students

Through the exploration of the results of the difference analysis of the dimensions of computational thinking before and after the experiments of the control group, we found that the conventional programming education activities have a certain enhancement effect on the computational thinking of college students, but its effect is not very significant. This is due to the fact that conventional programming education activities tend to focus only on learners' understanding and memorization of computational concepts such as sequence, condition, loop, operators, and other computational concepts in programming activities, as well as the operation and practice of computational practices such as writing programs, testing and debugging, which only remain

in the cultivation of the basic knowledge, and operational skills, neglecting the cultivation of learners' thinking such as decomposition, abstraction, induction, and summarization. Brennan et al. advocate the need to develop learners' computational thinking at both cognitive and non-cognitive levels to support this conclusion to some extent[29]. Nevertheless, because learning to program is a challenging and complex process, learners can become slack when they are unable to get timely guidance and help during the learning process. In our interviews, some learners in the control group said that "when faced with a complex programming project and did not know where to start, there was a certain degree of trepidation". This is a good indication that conventional programming education activities need to be further designed in order to effectively enhance learners' computational thinking. Zhang et al. have already attempted in this regard, as they enhanced college students' computational thinking, programming self-efficacy, and academic performance by providing learning scaffolds with progressive flowcharts in their programming education activities[30].

5.2 AIGC Tools in Programming Education Activities Significantly Enhance Computational Thinking in College Students

After a comprehensive analysis of the multidimensional differences in computational thinking before and after the experiments of the experimental group, it was found that the use of the AIGC tool in programming education activities had a significant effect on the enhancement of college students' computational thinking. Especially in key thinking such as decomposition, abstraction, and modeling, the improvement of evaluation ability is most prominent, but the improvement of the algorithm is not very significant.

Such results are related to the characteristics of the AIGC tool. Specifically, learners naturally exercise decomposition, abstraction, modeling, and evaluation as they work with the AIGC tool. For example, learners need to accurately describe the problem when asking questions to the AIGC tool for the AIGC tool to generate more accurate answers. To get accurate answers, learners sometimes need to break down complex questions into smaller questions that can be understood by AIGC. It is in this process of asking questions again and again that the learner's decomposition is subconsciously improved[24]. The difficulty in programming activities is the determination of program correctness and code optimization, which requires learners to repeatedly review the code they have written. The AIGC tool, with its ability to quickly locate erroneous code fragments and provide multiple optimization solutions, greatly reduces the time spent on single-time error location and iterative optimization, thus increasing the frequency of evaluation and optimization. This is the reason why the improvement in learner assessment is most prominent. The poor improvement in algorithmic ability is because the AIGC tool provides learners with direct answers without the process of transforming problem-solving from symbolic representations to computational models (sequences and rules).

5.3 Uncontrolled use of AIGC tools is Detrimental to Learners' Academic Achievement

Through the exploration of the results of the difference analysis of the academic performance between the experimental group and the control group after the test, we find that the AIGC tool facilitates the programming learning of the learners and reduces the requirements of the hands-on projects on the learners' basic knowledge of literacy and use, which makes the learners of the experimental group rely on the AIGC tool in the hands-on projects. It is difficult for them to complete various practical projects when they are separated from the support of the AIGC tool, which ultimately leads to the unsatisfactory academic performance of the experimental group. This is a strong indication that the uncontrolled use of AIGC tools in programming education activities is detrimental to the academic performance of learners.

In specific, during the practical training, learners in the experimental group will rely on the AIGC tool to complete a practical training project. In the process of practical training, learners can quickly complete the training projects by summarizing the answers based on the feedback from the AIGC tool. This process does not require learners to write and debug programs word by word, which reduces the frequency of learners' use of syntax, statements, and other basic knowledge, making most learners in the experimental group not very solid in their mastery of programming basics. This is the root cause of the poor performance of these learners in the academic achievement tests that are not supported by the AIGC tool. Moreover, some of the learners in the experimental group indicated that using the AIGC tool was about completing learning tasks and not about helping themselves learn. These learners immediately used the AIGC tool to get a solution to the problem at the first moment they encountered the problem, did not actively think about it, amplified learning inertia, and became carriers of the answers from the AIGC tool, which made them not know where to start when they independently faced the problems in the academic achievement test[31]. Therefore, to avoid AIGC tools replacing learners' thinking and over-reliance on AIGC tools, and to make AIGC tools useful learning assistants, there is a need to intensively monitor and manage learners' use of AIGC tools in learning activities.

6 Summary, Limitations, and Future Directions

The speedy development of Artificial Intelligence (AI) technology puts higher demands on the computational thinking of college students. However, the problem of lack of computational thinking is widespread among college students and adult learners at present. Therefore, there is an urgent need to find methods and measures for the cultivation of computational thinking among college students that have a wide range of applicability and good enhancement effects. Based on this, this study launched a 13-week educational activity of integrating AIGC tools into programming education with 225 freshmen students as research subjects. Based on analyzing and comparing the differences in learners' computational thinking and academic success before and after the experiment, and combining the results of interviews and exchanges, three conclusions were drawn. Based on the conclusions of this study, suggestions on how

to better utilize AIGC tools to enhance learning and promote the development of computational thinking ability are proposed, to provide certain references for the cultivation of college students' computational thinking. There are also some shortcomings in this study: the academic performance assessment focuses on programming knowledge and skills, and the dimensions of the assessment are not very comprehensive. Future research will focus on the behavior of learners and deeply explore the mechanism of the influence of AIGC tools on computational thinking. To better utilize AIGC tools to empower college students' computational thinking to enhance their achievements.

Fund Information

Funding: This work was supported by the National Education Science "14th Five-Year Plan" of the Ministry of Education Key Project "Artificial Intelligence Enabled Rural Teachers' Network Teaching and Research Mode Innovation Research" (Project No. DCA230459), and the Sichuan Center for Education Development Research Project of the year 2024"Research on Enrollment Strategies of Local Normal Universities under the Background of Examination and Enrollment System Reform"(Project No. CJF24069).

References

1. Reed, D.A., Bajcsy, R., Fernandez, M.A., Griffiths, J., Mott, R.D.: Computational science: Ensuring america's competitiveness.http://www.nitrd.gov/pitac/reports/index.html (2005).
2. Wing, J.M.: Computational thinking. Commun. ACM 49(3), 33–35 (2006) .
3. Voogt, J., Fisser, P., Good, J., Mishra, P., Yadav, A.: Computational thinking in compulsory education: Towards an agenda for research and practice. Education and Information Technologies 20(4), 715–728 (2015).
4. WANG You-mei, L.N.-y.Y.Y.-q.L.C.-c. NAN Xi-xuan: Programming resilience: A new issue for cultivating computational thinking in the digital age. Modern Educational Technology 33(02), 14–23 (2023).
5. Council, N.: Report of a workshop on the scope and nature of computational thinking. national academies press (2010).
6. FENG Youmei, L.X.Z.X. WANG Xinyi: What computational thinking is not: On the boundary of computational thinking and why it becomes the base of information technology discipline. e-Education Research 44(01), 84–90 (2023) .
7. National, C.U.S., Ebrary, I.: Report of a workshop on the pedagogical aspects of computational thinking. National Academies Press (2011).
8. Lyon, J.A., Magana, A.J.: Computational thinking in higher education: A review of the literature. Computer Applications in Engineering Education (2020).
9. GUO Ping, X.X. ZHUANG Wei: Teaching reform and practice of "principles of computer composition" for the cultivation of computational thinking ability under the background of artificial intelligence. Research and Exploration in Laboratory 42(12), 129–135 (2023).
10. FANG Min, C.W.L.S. SUN Ying: Current status and influencing factors of chinese preservice teachers' computational thinking: An empirical research on universities in tianjin, hebei, and shandong. Open Education Research 27(04), 98–110 (2021).

11. LIU Huibin, C.Q.H.R. HU Jianpeng: Four-dimensional extension of cultivation of college students' computational thinking ability based on multi-angular data analysis. Research and Exploration in Laboratory 41(12), 232–238 (2022).
12. LI Jiaying, K.C. LIU Na: The category characteristics and teaching enlightenment of computational thinking among college students. Journal of Jiangxi Normal University(Natural Science Edition) 48(02), 131–138 (2024).
13. Shen Shusheng, W.Z.: Research on scenario design to promote the cultivation of computational thinking in artificial intelligence course. Journal of Distance Education 41(06), 94–103 (2023).
14. Hsu, T.-C., Chang, C., Wu, L.-K., Looi, C.-K.: Effects of a pair programming educational robot-based approach on students' interdisciplinary learning of computational thinking and language learning. Frontiers in psychology 13, 888215 (2022)
15. Wang, Y., Xie, B.: Can robot-supported learning enhance computational thinking?—a meta-analysis. Thinking Skills and Creativity 52, 101528 (2024)
16. DUO Zhaojun, R.Y. LIU Yansong: Research on internal mechanism and teaching practice of programming education for development of children's computational thinking. e-Education Research 43(08), 101–108 (2022).
17. DUO Zhaojun, R.Y. LIU Yansong: Research on internal mechanism and teaching practice of programming education for development of children's computational thinking. e-Education Research 43(08), 101–108 (2022).
18. Zhang, L., Nouri, J.: A systematic review of learning computational thinking through scratch in k-9. Computers & Education 141, 103607 (2019)
19. Hsu, T.-C., Chang, S.-C., Hung, Y.-T.: How to learn and how to teach computational thinking: Suggestions based on a review of the literature. Computers & Education 126, 296–310 (2018)
20. Sun Lihui, H.L.: Can programming really promote the individual development of children? a meta-analysis of 28 experimental and quasi-experimental studies. Journal of East China Normal University(Educational Sciences) 39(11), 45–58 (2021).
21. Lund, B.D., Wang, T.: Chatting about chatgpt: how may ai and gpt impact academia and libraries? Library hi tech news 40(3), 26–29 (2023)
22. Yang Zongkai, W.D.C.X. Wang Jun: Exploring the impact of chatgpt/aigc on education and strategies for response. Journal of East China Normal University(Educational Sciences) 41(07), 26–35 (2023).
23. DONG Yan, L.X.H.Y. XIA Liangliang: Analysis of the path to empowering student learning with chatgpt. e-Education Research 44(12), 14–2034 (2023).
24. SUN Dan, X.Z.X.G. ZHU Chengcong: A study on analysis of college students'programming learning behavior based on generative artificial intelligence. e-Education Research 45(03), 113–120 (2024).
25. Chen Xingye, M.Y.: Construction and exploration of evaluation index system of localized computational thinking:based on the sample analysis and verification of 1410 senior high school students. Journal of Distance Education 38(05), 70–80 (2020).
26. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada, vol. 1, p. 25 (2012)
27. ZHANG Yi, F.W.L.J.L.Y.D.S. CHEN Dengkang: A study on the construction of computational thinking scale for primary and secondary school students based on new curriculum standards. e-Education Research 45(03), 90–98 (2024).

28. Kang, C., Liu, N., Zhu, Y., Li, F., Zeng, P.: Developing college students' computational thinking multidimensional test based on life story situations. Education and Information Technologies 28(3), 2661–2679 (2023)
29. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada, vol. 1, p. 25 (2012)
30. Zhang, J.-H., Meng, B., Zou, L.-C., Zhu, Y., Hwang, G.-J.: Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy. Interactive Learning Environments 31(6), 3792–3809 (2023)
31. Zheng Yonghe, Z.Y.T.X.W.J.Z.Y. Zhou Danhua: Chatgpt from the perspective of computational education: Connotation, theme, reflection, and challenge. Journal of East China Normal University(Educational Sciences) 41(07), 91–102 (2023).