Curriculum and Teaching Methodology (2026)
Clausius Scientific Press, Canada

DOI: 10.23977/curtm.2026.090403
ISSN 2616-2261 Vol. 9 Num. 4

Research on the Mechanism and Path of AIGC Empowering Junior High School Students' Programming Learning

Danhong Huang*, Xianqin Wang, Lei Lei

School of Computer and Information Science, Chongqing Normal University, Chongqing, 401331, China

**Corresponding Author*

Keywords: AIGC; generative artificial intelligence; junior high school students; programming learning; information technology education; computational thinking; human-machine collaboration

Abstract: AIGC is reshaping programming education by providing generative resources, conversational guidance, code feedback, and adaptive learning support. Junior high school students, however, often face difficulties in understanding abstract programming concepts, transferring algorithmic thinking to real problems, and maintaining learning autonomy. Based on literature analysis and conceptual synthesis, this paper examines the mechanism and path through which AIGC empowers programming learning in junior high school information technology education. The study finds that AIGC empowerment operates through five interrelated mechanisms: cognitive scaffolding, dialogic feedback, human-machine collaboration, creative generation, and data-driven evaluation. Accordingly, it proposes a developmental path integrating core-literacy-oriented objectives, staged programming activities, project-based learning, personalized support, teacher-guided human-machine interaction, and ethical governance. The study contributes a systematic framework for transforming AIGC from a technical tool into an instructional support system for cultivating computational thinking, programming ability, creativity, and responsible digital citizenship.

1. Introduction

The rapid development of artificial intelligence has accelerated the digital transformation of basic education and has created new possibilities for programming learning in junior high school information technology courses. In the context of curriculum reform, junior high school information technology teaching is no longer limited to software operation or knowledge transmission; instead, it increasingly emphasizes information awareness, computational thinking, digital learning and innovation, and information social responsibility. Existing research on AI-enabled junior high school information technology teaching has shown that intelligent technologies can support learning diagnosis, resource generation, immersive interaction, and process-based evaluation, thereby promoting the intelligent transformation of classroom teaching [1]. This shift provides an important

background for examining how AIGC can support students' programming learning.

Programming learning is a key carrier for cultivating computational thinking. It requires learners to analyze problems, design algorithms, express solutions through code, test programs, and revise outputs through reflection. However, programming concepts such as variables, loops, conditions, functions, and algorithms are abstract for many junior high school students. Generative artificial intelligence has been found to provide code examples, debugging suggestions, optimization ideas, and personalized guidance, which may help learners develop computational thinking through deeper interaction [2]. Therefore, the educational value of AIGC lies not merely in generating code, but in supporting students' progressive cognitive development and problem-solving processes.

Current studies have explored AI-empowered programming education from multiple perspectives. Research on AI-enabled programming course innovation emphasizes student-centered design, project-based learning, digital resources, and diversified formative assessment [3]. Studies on generative AI-supported project-based learning in junior high school information technology show that AIGC can support project initiation, planning, implementation, management, and evaluation [4]. Empirical research on high school programming teaching indicates that GenAI can improve students' programming confidence, program understanding, situational problem-solving knowledge, code implementation, and innovative design ability, although its effects on basic concepts and interest may be limited [5]. These findings suggest that AIGC has practical potential but must be embedded in an appropriate instructional structure.

Other studies provide additional support for constructing such a structure. The DeepSeek-supported Python teaching model highlights the integration of teachers, students, and intelligent tools, arguing that students should interact with AI in authentic programming tasks while teachers act as guides and motivators [6]. Junior high school classroom cases show that AI can assist lesson preparation, interactive tasks, algorithm games, and unit practice, thereby transforming static knowledge learning into dynamic creation [7]. Digital programming classroom studies further emphasize the role of learning platforms, online development environments, AI programming assistants, automated testing, and learning data analysis [8]. Research on JavaScript programming instruction demonstrates that AI tools can support code completion, code review, interface prototyping, and individualized feedback [9]. At the same time, studies on classroom application strategies warn that teachers must balance group and individual needs, learning interest and sustained engagement, preset design and generative learning, human-machine interaction effectiveness, information authenticity, and privacy protection [10]. Research on program design course reform also suggests that large language models can be integrated into pre-class, in-class, and post-class activities, but students should maintain independent thinking and use AI as an auxiliary tool rather than a substitute [11].

Despite these studies, three research gaps remain. First, many studies focus on high school, vocational education, or higher education, while the junior high school stage has received comparatively less systematic attention. Second, existing studies often discuss teaching strategies or practical cases, but the internal mechanism by which AIGC supports junior high school programming learning needs further clarification. Third, there is still a lack of an integrated path that connects conceptual definition, learning dilemmas, empowerment mechanism, instructional design, and implementation guarantee. Therefore, this paper focuses on the question: how can AIGC empower junior high school students' programming learning, and through what mechanisms and paths can such empowerment be realized effectively?

This paper adopts literature-based conceptual analysis and design-oriented reasoning. It first defines the core concepts of AIGC empowerment and junior high school programming learning. It then analyzes the existing developmental dilemmas of programming learning. On this basis, it constructs the internal mechanism of AIGC empowerment and proposes a multi-dimensional path

for instructional implementation. Finally, it discusses the study's innovation, limitations, and future research directions.

2. Core Conceptual Definitions of AIGC Empowering Junior High School Students' Programming Learning

2.1 Definition of AIGC in Programming Learning

AIGC refers to artificial-intelligence-generated content created through large models and related intelligent systems. In programming learning, AIGC includes natural-language explanation, code generation, error diagnosis, algorithmic suggestion, interactive questioning, resource recommendation, multimodal task presentation, and personalized feedback. Its educational significance does not depend solely on whether it can generate correct answers. More importantly, it can participate in the learning process as a cognitive tool, feedback provider, creative partner, and reflective support.

For junior high school programming learning, AIGC should be understood as a learning-support environment rather than a code-production machine. If students directly copy AI-generated code, the tool may weaken their understanding and reduce opportunities for thinking. If teachers design appropriate tasks, prompts, constraints, and reflection activities, AIGC can help students clarify problem requirements, decompose complex tasks, test possible solutions, and revise programs through evidence. The essence of AIGC empowerment is therefore the reconstruction of the relationship among learning resources, programming tasks, teacher guidance, student inquiry, and evaluation feedback.

2.2 Definition of Junior High School Programming Learning

Junior high school programming learning refers to the process in which students learn to understand computational concepts, express algorithms, design programs, debug errors, and solve real or simulated problems through programming languages, visual tools, or text-based coding environments. Its objective is not only to enable students to write simple programs, but also to cultivate computational thinking, logical reasoning, problem decomposition, abstraction, testing awareness, collaboration, creativity, and responsible technology use.

Compared with programming learning in high school or higher education, junior high school programming learning has distinctive developmental characteristics. Students are at a stage where abstract thinking is developing but still requires concrete scenarios, intuitive demonstrations, and step-by-step scaffolding. They may show strong curiosity toward interactive and creative tasks, but their persistence and metacognitive regulation are still limited. Therefore, programming instruction at this stage should avoid excessive syntax explanation and should instead connect basic concepts with meaningful problems, visible outputs, and guided reflection.

2.3 Definition of AIGC Empowerment

AIGC empowerment in programming learning means using generative AI to enhance the quality, adaptability, interaction, creativity, and evaluability of programming instruction. It includes at least five dimensions. The first is resource empowerment, through which AIGC generates examples, micro-tasks, explanations, and project materials. The second is cognitive empowerment, through which AIGC helps students understand concepts, decompose problems, and compare solution strategies. The third is process empowerment, through which AIGC provides timely feedback in coding, debugging, and revision. The fourth is creative empowerment, through which AIGC

inspires students to design original works and extend programming tasks to authentic contexts. The fifth is evaluation empowerment, through which AIGC assists teachers in collecting learning evidence and providing formative feedback.

This definition stresses that empowerment is not technological replacement. Teachers remain responsible for instructional objectives, task design, value guidance, ethical regulation, and final evaluation. Students remain the main agents of thinking, decision-making, coding, debugging, and reflection. AIGC is effective only when it is embedded in a pedagogically meaningful learning structure.

3. Existing Developmental Dilemmas in Junior High School Students' Programming Learning

3.1 Uneven Learning Foundations and Insufficient Individualized Support

Junior high school students differ greatly in prior experience, logical reasoning, reading comprehension, typing ability, and familiarity with digital tools. In the same classroom, some students may quickly understand algorithmic logic, while others may struggle with basic syntax and operational steps. Traditional whole-class teaching often follows a unified pace, making it difficult to provide differentiated support. Students with stronger foundations may find repetitive exercises unchallenging, while students with weaker foundations may gradually lose confidence.

AI-enabled junior high school information technology teaching emphasizes real-time learning data collection, diagnosis of weak points, and individualized resource generation [1]. This provides a possible solution to the contradiction between large-class teaching and individual differences. However, if AIGC is used only as an answer generator, it may further enlarge learning differences, because students with stronger self-regulation can use it productively, while students with weaker foundations may become dependent on generated results. Therefore, individualized support must be accompanied by teacher-designed scaffolds and learning rules.

3.2 Syntax-Centered Teaching and Weak Problem Orientation

Programming classes sometimes overemphasize grammar explanation and mechanical imitation. Students may learn how to write a loop or conditional statement, but they may not understand why such structures are needed in problem solving. As a result, their knowledge remains fragmented, and they find it difficult to transfer programming concepts to real situations. Research on programming course innovation points out that traditional methods often fail to integrate programming skills with professional or real-life contexts and therefore need project-based and intelligent reform [3].

For junior high school students, programming should be linked with familiar problems such as campus services, health data, simple games, environmental monitoring, digital storytelling, and information visualization. AIGC can generate contextualized tasks and help students compare multiple solutions, but the teacher must ensure that the task remains appropriate to students' cognitive level. The purpose is to make students experience programming as a means of problem solving rather than a list of isolated rules.

3.3 Delayed Feedback and Low Debugging Efficiency

Debugging is central to programming learning. However, students often experience frustration when they encounter syntax errors, logical bugs, or unexpected outputs. In traditional classrooms, teachers cannot respond instantly to every student's problem. Some students wait passively, while others repeatedly try without understanding the cause of the error. This reduces learning efficiency

and weakens programming confidence.

AIGC can provide immediate explanations, error localization, and revision suggestions. Studies on GenAI-supported programming learning indicate that code examples, debugging suggestions, and optimization feedback can improve learners' mastery of programming skills and support computational thinking development [2]. Nevertheless, feedback must be carefully controlled. If AI directly provides complete corrected code, students may skip the process of reasoning. Effective feedback should guide students to identify the error type, explain the cause, predict the output, and revise the code independently.

3.4 Weak Connection between Knowledge Learning and Creative Practice

Programming learning should ultimately lead to creative application. Yet many classroom tasks remain small, closed, and answer-oriented. Students complete exercises but seldom design meaningful products. This limits the development of innovation, collaboration, and authentic problem-solving ability. Generative AI-supported project-based learning has shown that AIGC can support project planning, resource generation, team division, and product improvement [4]. Junior high school practice cases also show that AI can transform writing, programming, and interactive design into integrated project activities [7].

The key problem is how to move from tool use to project creation. Students should not simply ask AI to produce a project. Instead, they should define a problem, propose an initial idea, use AIGC to expand possibilities, evaluate feasibility, implement core functions, test outputs, and reflect on improvement. Only in this process can AIGC support genuine creative learning.

3.5 Risks of Dependence, Inaccuracy, and Ethical Misuse

AIGC also brings risks. Generated code may contain hidden errors, inefficient logic, or inappropriate solutions. Generated explanations may be plausible but inaccurate. Students may over-rely on AI, submit AI-generated code without understanding it, or neglect academic integrity. Studies on classroom application strategies emphasize the need to verify generated content, protect privacy, and cultivate responsible use [10]. Program design course reform also warns that students should keep independent thinking and treat intelligent tools as assistance rather than replacement [11].

For junior high school students, these risks are more significant because their critical judgment and ethical awareness are still developing. Therefore, AIGC-based programming learning must include AI literacy, prompt norms, code verification, source awareness, privacy protection, and responsibility education.

4. Internal Mechanism of AIGC Empowering Junior High School Students' Programming Learning

4.1 Cognitive Scaffolding Mechanism

The first mechanism is cognitive scaffolding. Programming requires students to move from concrete situations to abstract structures. AIGC can help learners translate natural-language problems into computational representations, identify input and output, decompose tasks, clarify variables, and design algorithmic steps. This corresponds to the development of computational thinking dimensions such as decomposition, abstraction, algorithmic design, evaluation, and generalization.

For junior high school students, cognitive scaffolding should be progressive. At the beginning

stage, AIGC may provide concept explanations, analogies, and simple examples. At the guided practice stage, it may ask questions that lead students to identify conditions, loops, and data relationships. At the independent creation stage, it may help students compare alternative algorithms or optimize program structures. The model of GenAI-supported programming learning emphasizes staged roles such as intelligent companion, intelligent mentor, intelligent evaluation assistant, and intelligent innovation engine [2]. This provides theoretical support for assigning different scaffolding functions according to students' learning stages.

4.2 Dialogic Feedback Mechanism

The second mechanism is dialogic feedback. Unlike static textbooks or delayed teacher comments, AIGC can interact with students through multiple rounds of questioning and response. Students can ask why an error occurs, request a simpler explanation, compare two solutions, or test whether their reasoning is correct. This interaction can transform feedback from a final judgment into a continuous learning process.

However, dialogic feedback must be designed to promote thinking. Teachers can require students to use prompts such as "Explain the error without giving the full answer," "Ask me three questions to help me find the bug," or "Compare my algorithm with another method and point out the difference." In this way, AIGC feedback becomes a metacognitive scaffold. Students learn not only how to correct code, but also how to evaluate their own reasoning and regulate their learning process.

4.3 Human-Machine Collaborative Mechanism

The third mechanism is human-machine collaboration. AIGC changes the classroom interaction structure from teacher-student dual interaction to teacher-student-machine triadic interaction. In this structure, the teacher designs the learning task, sets the boundaries of AI use, organizes discussion, and evaluates learning evidence. The student explores, codes, questions, verifies, and reflects. AIGC provides generative support, adaptive feedback, and resource expansion.

The DeepSeek-supported Python teaching model stresses the integration of teachers, students, and intelligent systems, with teachers serving as guides and students interacting with AI in programming tasks [6]. This insight is particularly relevant to junior high school programming learning. Students need AI support, but they also need teacher guidance to avoid superficial interaction. Human-machine collaboration should therefore follow the principle of teacher-led, student-centered, and AI-assisted learning.

4.4 Creative Generation Mechanism

The fourth mechanism is creative generation. AIGC can generate diverse ideas, scenarios, interface drafts, project plans, test data, and improvement suggestions. This helps students move beyond imitation and engage in creative programming. For example, in a project on designing a health assistant, students may ask AIGC to suggest possible functions, then choose a feasible function such as data input, conditional judgment, or simple visualization. They can then implement the core logic themselves and use AI to refine the interface or explanation.

Studies on junior high school classroom practice show that AI-supported activities can transform static content into interactive programming projects and enhance students' innovation potential [7]. Research on JavaScript teaching also suggests that AI-assisted prototyping and code review can reduce non-core burdens and allow students to focus on the main programming objectives [9]. Therefore, AIGC can stimulate creative thinking when it is used to support ideation and iteration

rather than replace students' core coding process.

4.5 Data-Driven Evaluation Mechanism

The fifth mechanism is data-driven evaluation. Programming learning leaves rich process traces, including code versions, error types, debugging time, prompt records, peer comments, and final works. AIGC and digital platforms can help collect and analyze these data to support formative assessment. AI-enabled programming teaching studies have highlighted the value of diversified process assessment, learning behavior analysis, progress tracking, and intelligent evaluation [3]. Digital programming classroom research also emphasizes learning platforms, automated testing, and student data analysis [8].

In junior high school programming learning, data-driven evaluation should include both technical and thinking dimensions. Teachers can evaluate whether students understand concepts, decompose problems, test programs, respond to feedback, collaborate with peers, and use AI responsibly. Evaluation should not focus only on whether the final code runs. It should also examine the learning process and the quality of human-machine interaction.

5. Multiple Development Paths for AIGC Empowering Junior High School Students' Programming Learning

5.1 Reconstructing Learning Objectives around Core Literacy

The first path is to reconstruct programming learning objectives around core literacy. AIGC-supported programming learning should not aim merely at faster code production. Its objectives should include conceptual understanding, computational thinking, problem-solving ability, digital creation, collaboration, and ethical responsibility. This is consistent with junior high school information technology teaching reform, which emphasizes core-literacy-oriented intelligent objectives [1].

Teachers can design objectives at three levels. The first level is knowledge and skill, including understanding variables, conditions, loops, functions, and simple algorithms. The second level is process and method, including problem decomposition, code testing, debugging, and reflection. The third level is attitude and responsibility, including curiosity, persistence, academic integrity, AI verification awareness, and data privacy awareness. AIGC should be mapped to these objectives rather than used as an isolated classroom novelty.

5.2 Designing a Staged Programming Learning Activity System

The second path is to design staged activities. Based on the cognitive characteristics of junior high school students, programming learning can be organized into five stages: guided perception, concept construction, assisted coding, collaborative debugging, and creative transfer.

In the guided perception stage, students encounter a real problem and describe it in natural language. AIGC can help generate examples or questions that activate prior knowledge. In the concept construction stage, students learn key programming structures through demonstrations and guided prompts. In the assisted coding stage, students write partial programs and use AIGC for hints rather than full answers. In the collaborative debugging stage, students compare code, identify errors, and ask AIGC to explain possible causes. In the creative transfer stage, students redesign or extend the task to produce a small digital work. This staged design draws on the idea that GenAI can support different learning phases through different roles [2].

5.3 Integrating AIGC with Project-Based Programming Learning

The third path is to integrate AIGC with project-based learning. AIGC is particularly suitable for project learning because it can support topic selection, plan formulation, resource search, task division, product iteration, and reflective evaluation. In junior high school programming courses, projects should be small, authentic, feasible, and connected with students' life. Possible projects include campus lost-and-found systems, simple quiz games, environmental data displays, physical health calculators, digital story generators, and class service tools.

AIGC-supported project-based learning should follow a clear process. Students first identify a problem and propose an initial solution. They then use AIGC to expand ideas, but must evaluate feasibility. Next, they divide tasks, design the program structure, implement core functions, test results, revise the product, and present the final work. Research on generative AI-supported junior high school project-based learning provides practical evidence that AIGC can enhance resource generation, personalized guidance, and group collaboration [4]. The key is to require students to explain their design choices and demonstrate their own contribution.

5.4 Building Personalized and Tiered Learning Support

The fourth path is to build personalized and tiered support. Teachers can use learning data and classroom observation to divide programming tasks into basic, advanced, and creative levels. Basic tasks ensure that all students understand essential concepts. Advanced tasks help students apply concepts to more complex situations. Creative tasks allow students to design original extensions.

AIGC can provide different forms of support for different students. For beginners, it can explain concepts in simple language, offer step-by-step hints, and generate practice questions. For intermediate learners, it can compare algorithms, suggest debugging strategies, and provide code-style feedback. For advanced learners, it can propose optimization challenges, creative extensions, and interdisciplinary applications. Studies on AI-enabled programming teaching confirm that personalized learning recommendation, knowledge graphs, and learning-path analysis can help teachers understand students' learning states and provide targeted guidance [11].

5.5 Constructing a Teacher-Guided Human-Machine Interaction Model

The fifth path is to construct a teacher-guided interaction model. Junior high school students need explicit norms for interacting with AIGC. Teachers can teach students how to ask clear questions, how to describe errors, how to request hints, how to verify generated answers, and how to record the learning process.

A practical interaction model may include four steps: ask, judge, revise, and reflect. In the "ask" step, students formulate a prompt based on the problem. In the "judge" step, they evaluate whether the response is accurate and suitable. In the "revise" step, they modify their code or idea according to evidence. In the "reflect" step, they explain what they learned and what remains uncertain. Digital programming classroom studies have shown that AI programming assistants, online development environments, and automated feedback can support such learning when combined with platform-based resources and teacher guidance [8].

5.6 Embedding AIGC Use in Ethical and Responsible Learning

The sixth path is to embed AIGC use in ethical and responsible learning. Students should understand that AI-generated content may be wrong, biased, or incomplete. They should learn to protect personal information, avoid uploading sensitive data, respect intellectual property, and

distinguish assistance from plagiarism.

Teachers can establish classroom rules such as: AI may be used for explanation, hints, debugging questions, and idea expansion; AI may not be used to submit complete work without understanding; all AI-assisted parts should be explained by the student; and students must verify generated code before use. These rules correspond to the application principles proposed in information technology classroom strategy research, including authenticity verification, effective human-machine interaction, privacy protection, and balanced support for group and individual learning [10].

6. Implementation Guarantee Strategies

6.1 Establishing an Integrated Digital Learning Platform

AIGC empowerment requires a stable platform environment. Schools should provide accessible programming environments, AI interaction interfaces, resource repositories, assignment systems, and data dashboards. The platform should support code writing, execution, debugging, prompt recording, teacher feedback, peer review, and learning analytics. For junior high school classrooms, the platform should be simple, secure, and age-appropriate. It should avoid unnecessary login barriers and reduce technical distractions.

6.2 Improving Teachers' AIGC Pedagogical Competence

Teachers' competence is the key guarantee. Teachers need to understand not only how to operate AIGC tools, but also how to transform them into learning supports. Their competence should include prompt design, task design, AI response evaluation, classroom regulation, data interpretation, ethical guidance, and differentiated instruction. Studies on AI-supported programming education repeatedly show that the teacher remains central in selecting content, designing activities, interpreting data, and maintaining student engagement [6][11].

Professional development should therefore move beyond tool demonstration. It should include lesson design workshops, classroom case analysis, action research, and collaborative reflection. Teachers can jointly develop prompt libraries, debugging cases, project templates, evaluation rubrics, and ethical-use guidelines.

6.3 Developing Process-Oriented and Multi-Dimensional Evaluation

Evaluation should measure the whole programming learning process. A multi-dimensional rubric may include concept understanding, problem decomposition, algorithm design, code implementation, debugging strategy, interaction with AIGC, collaboration, creativity, and responsible use. Students can submit not only final code, but also design notes, prompt records, debugging logs, peer feedback, and reflection reports.

Empirical research on GenAI-supported programming teaching reminds us that AIGC may significantly improve some dimensions, such as programming confidence and code implementation, but may have weaker effects on basic concept mastery or interest [5]. Therefore, evaluation should help teachers identify which dimensions are improved and which still require direct instruction.

6.4 Creating a Safe and Sustainable Governance System

Finally, schools need governance mechanisms. These include tool selection standards, privacy protection measures, age-appropriate access control, academic integrity rules, teacher review

procedures, and emergency plans for incorrect or inappropriate AI outputs. AIGC governance should not simply restrict use; it should help students learn how to use intelligent tools responsibly. Sustainable implementation also requires collaboration among school administrators, information technology teachers, parents, and platform providers.

7. Discussion

7.1 Innovation of This Study

Compared with existing studies, this paper makes three contributions. First, it focuses specifically on junior high school programming learning. Existing research has provided valuable evidence from higher education, vocational education, and high school contexts, but junior high school students have distinctive cognitive, motivational, and ethical-development characteristics. This paper adapts AIGC empowerment to their learning needs.

Second, the paper constructs an integrated mechanism framework. Instead of treating AIGC as a general tool, it identifies five mechanisms: cognitive scaffolding, dialogic feedback, human-machine collaboration, creative generation, and data-driven evaluation. These mechanisms explain how AIGC can support programming learning from concept understanding to creative application.

Third, the paper proposes a path system connecting objectives, activities, projects, personalization, interaction, ethics, and implementation guarantees. This responds to the limitation that some existing discussions focus on isolated strategies without forming a coherent instructional ecology.

7.2 Limitations

This study also has limitations. First, it is a conceptual and literature-based study rather than an empirical intervention. Although it draws on existing research and classroom cases, the proposed path still needs to be tested through classroom experiments, action research, or quasi-experimental studies. Second, the paper mainly considers the Chinese junior high school information technology curriculum context. Its conclusions may need adaptation in different educational systems, programming languages, and school conditions. Third, AIGC technologies are developing rapidly. The functions, risks, and classroom feasibility of specific tools may change over time, so the framework requires continuous revision.

7.3 Future Research

Future research can proceed in three directions. First, researchers can design and implement AIGC-supported junior high school programming units and examine their effects on computational thinking, programming achievement, learning motivation, and self-efficacy. Second, future studies can develop evaluation instruments for human-machine collaborative programming learning, including prompt quality, debugging reasoning, AI verification behavior, and ethical awareness. Third, longitudinal research can explore whether AIGC-supported learning improves students' sustained interest in programming and their ability to transfer computational thinking to interdisciplinary problem solving.

8. Conclusion

AIGC provides important opportunities for improving junior high school students' programming learning, but its value depends on pedagogical design rather than technological novelty. This paper

defines AIGC empowerment as a systematic process involving resource generation, cognitive support, dialogic feedback, creative stimulation, and learning evaluation. It identifies the main dilemmas of junior high school programming learning, including uneven foundations, syntax-centered teaching, delayed feedback, weak creative transfer, and ethical risks. It then constructs five empowerment mechanisms: cognitive scaffolding, dialogic feedback, human-machine collaboration, creative generation, and data-driven evaluation.

On this basis, the paper proposes a multi-dimensional development path: reconstructing learning objectives around core literacy, designing staged programming activities, integrating AIGC with project-based learning, building personalized and tiered support, constructing teacher-guided human-machine interaction, and embedding ethical responsibility into learning. To ensure implementation, schools should establish integrated digital platforms, improve teachers' AIGC pedagogical competence, develop process-oriented evaluation, and create safe governance systems.

In essence, AIGC should not replace students' thinking or teachers' guidance. Its proper role is to expand learning resources, make feedback more timely, support individualized exploration, stimulate creative programming, and provide evidence for formative evaluation. When used responsibly and systematically, AIGC can help junior high school programming education move from code imitation to problem solving, from fragmented knowledge to computational thinking, and from tool operation to innovative digital creation.

References

[1] HOU Yu. Practice of Artificial Intelligence Empowering Junior High School Information Technology Teaching[J]. China New Telecommunications,2026,28(8):110-112.

[2] ZHANG Jie, ZHAI Haiqi, ZHANG Jing. The Mechanisms and Pathways of Generative AI Empowering Programming Education[J]. E-Education Research,2026,47(4):64-71.DOI:10.13811/j.cnki.eer.2026.04.008.

[3] ZHANG Xiying, REN Zhikao, YE Chen, et al. Innovative Research and Application of AI-Enabled Programming Course Teaching[J/OL]. Journal of Jiangsu Vocational Institute of Architectural Technology,1-8[2026-07-03]. https://doi.org/10.19712/j.cnki.jsjyxb.20251222.001.

[4] YUAN Feixue, SHANG Lele, DUAN Yuanmei. Design and Implementation of Generative Artificial Intelligence Empowering Project-Based Learning in Junior High School Information Technology: Taking "Creative Promotion of the Internet World" as an Example[J]. China Modern Educational Equipment,2025,(18):30-32. DOI:10.13492/j.cnki.cmee.2025.18.017.

[5] ZHAO Luyao, YANG Xianmin, LAN Ying. Research on the Effectiveness of Generative Artificial Intelligence Empowering High School Programming Teaching[J]. Modern Educational Technology,2025,35(9):56-65.

[6] QU Yuwei, ZHANG Chunlan. Exploration of a Teacher-Student-Machine Integrated Teaching Model for Python Programming Courses Empowered by DeepSeek-like Artificial Intelligence[J]. Modern Vocational Education, 2025,(24):85-88.

[7] XU Hui. Artificial Intelligence Empowering Junior High School Information Technology Classroom Teaching: A Deep Integration of Wisdom and Creativity[J]. Shandong Education,2025,(Z5):28-30.

[8] LIN Zhuorui. Exploration of AI-Empowered Digital Programming Classroom in Senior High School[J]. Computer Education,2025,(4):27-32.DOI:10.16512/j.cnki.jsjjy.2025.04.006.

[9] CHEN Hanshen. Teaching Programming Courses Empowered by Artificial Intelligence: Taking JavaScript Programming Course as an Example[J]. Computer Knowledge and Technology,2024,20(28):137-139. DOI:10.14004/j.cnki.ckt.2024.1496.

[10] GU Wenjie, DING Jie. Application Strategies of Generative Artificial Intelligence Empowering Information Technology Classroom Teaching[J]. China Information Technology Education,2024,(17):95-99.

[11] GAO Honghao, CHEN Zhangjin. Teaching Reform of Programming Courses Empowered by Artificial Intelligence[J]. Computer Education, 2024,(7):41-43+48.DOI:10.16512/j.cnki.jsjjy.2024.07.029.