



# Digital Transformation of education: the impact of generative AI on programmer training

Egor CHEGLOV<sup>1</sup>, Andrei PESTUNOV<sup>2</sup>, Olga SHVETS<sup>3</sup>

**Abstract:** With the development of generative artificial intelligence (AI), students are increasingly using it to solve programming tasks. This leads to changes in the educational process and requires revision of traditional teaching methods. The study analysed the patterns of AI usage, its advantages and disadvantages, and identified key differences between code written manually and generated by neural networks. Possible approaches to the integration of AI into the learning system are considered in order to increase the effectiveness and ensure the objectivity of knowledge assessment.

**Keywords:** code generation; digital transformation; generative AI; knowledge assessment; programmer training; programming education; teaching methods integration.

## 1 INTRODUCTION

With the advancement of generative artificial intelligence (AI), students have gained access to computational tools that facilitate the automation of academic tasks across various disciplines. The 2023 study conducted by researchers from Darmstadt University of Applied Sciences (Germany), sampling over 8,800 students, examines the prevalence and domains of AI in education. The findings indicate that more than 63% of respondents incorporate AI-driven technologies into their learning processes. The adoption rate is notably higher among students in engineering disciplines (75%), followed by those in the natural sciences (71%), humanities (61%), and medical fields (52%) [1].

Programming is one of the academic disciplines most closely associated with AI. The integration of AI into programming education presents both advantages and disadvantages [2]. Generative AI facilitates the development of solutions by autonomously producing accurate and comprehensible program code, thereby significantly reducing the time and computational resources required to complete assignments. However, the incorporation of generative AI into the educational process has sparked extensive debate among educators and researchers.

On the one hand, AI can function as a personalized assistant for students, providing explanations of algorithms and guiding the development of complex software structures, as well as systematically organizing and presenting information in an accessible format [3]. On the other hand, concerns have been raised that relying on generative AI may diminish students' motivation for independent problem-solving and impede the advancement of critical and creative thinking skills, potentially leading to a decline in overall knowledge acquisition [4].

This raises several critical questions: does the use of generative artificial intelligence (AI) pose a threat to the educational process? Can it facilitate a deeper understanding of complex topics by freeing up time for analysis and creativity, or, conversely, does it deprive students of essential hands-on experience?

The objective of this study is to examine students' experiences in utilizing generative AI for solving programming-related learning tasks. The research aims to identify the

distinctive characteristics and patterns of AI usage among students from various academic backgrounds, thereby providing a more comprehensive understanding of how AI integration influences the educational process in the context of programming instruction.

Additionally, this study includes an analysis of an electronic programming course hosted on the university's Learning Management System (LMS). Universities commonly employ the CodeRunner plugin for the automated assessment of students' solutions [5].

Traditional teaching process often assumes that a teacher plays both roles: teaching and checking. In such a situation, the second (checking) role may dominate over the first one and the student may fall into some sort of confrontation with the teacher. Automatic systems such as CodeRunner take the checking role by themselves and the teacher becomes an ally with the student. Their common goal is to overcome all checking barriers of the automatic system. The teacher doesn't help his student directly, but tries to help him (her) to handle the problem.

However, GAI (generative AI) may spoil these educational process model by creating temptation of exploiting it in order to solve the problem with the help of GAI instead of the teacher. There exists a difference between the teacher's and the GAI's help. The teacher is able to provide the student with the partial solution and the student would try to reach the final one. But GUI will elaborate the final solution at once and the student will be satisfied with that, thus, eliminating the teaching effect.

Thus, the study explores the challenge of detecting generative AI usage in academic assignments and proposes methods for identifying such cases within an automated verification environment.

## 2 RESEARCH METHODOLOGY

This study investigates the perceptions and experiences of students utilizing various generative AI models to complete programming tasks. To facilitate data collection, a structured questionnaire was designed, combining both quantitative measures and open-ended questions to capture qualitative insights.

The survey was administered to students enrolled in programming-related disciplines, including "Fundamentals of

Algorithmizing and Programming" (secondary vocational education), as well as "Programming," "Programming Languages," and "Programming of Discrete Structures" (bachelor's degree programs). Participation was voluntary and anonymous. A total of 61 students participated, comprising 29 students from secondary vocational education programs and 32 students from bachelor's degree programs.

In addition, the study incorporated 100 tasks of basic complexity from an electronic course, organized into eight thematic blocks. The experimental procedure involved simulating the behaviour of a student who copies the task description from the LMS and inputs it into a generative AI model. The AI-generated solution is then submitted to an automated testing system for verification without modification. In case of failure, the student iterates the process by submitting clarifying queries to the AI model until the solution passes the tests successfully.

### 3 SURVEY RESULTS

According to the results of the survey, 34 students (55.7%) reported possessing only basic programming knowledge, while 11 students (18%) indicated that they were learning programming for the first time. Another 11 respondents (18%) self-assessed their skills as confident in one programming language, and only 5 students (8.2%) stated that they are proficient in multiple programming languages.

In their self-assessment of algorithmic knowledge, 47.5% of respondents acknowledged a lack of familiarity with this area, while 42.6% of respondents indicated that they were acquainted with basic algorithms, such as brute force or bubble sort. Only 6 students (9.8%) reported proficiency in more advanced algorithms, such as binary search, quick sort, or Euclid's algorithm.

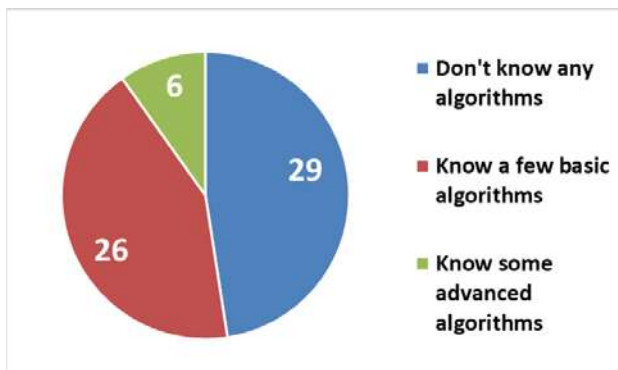


Figure 1 How would you rate your algorithms knowledge?

An analysis of the responses revealed two distinct groups based on the students' level of proficiency in programming and algorithms. Among those with limited or basic programming knowledge, the use of generative AI was most prevalent due to challenges faced after several unsuccessful attempts to complete tasks independently (67%) and the need to debug their own code (65%). In contrast, students with more advanced programming skills in one or more languages were more likely to turn to AI due to poor understanding of task conditions or the meaning of the problem (44%) and a lack of motivation to solve the task independently (38%). Additionally, both groups expressed a desire to obtain optimized solutions (13.3% of the total respondents).

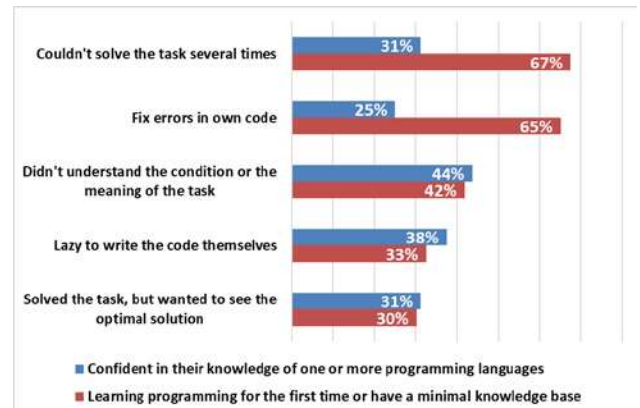


Figure 2 What is your main reason for using generative AI?

An analysis of the frequency of AI usage revealed that 73.8% of the respondents use it 1–3 times per week, 16.4% use it daily, and 9.8% use it 4–5 times per week.

The students identified several significant advantages of using generative AI in programming tasks. The most frequently mentioned benefits included time-saving, reported by 82.2% of respondents, and assistance in mastering new and complex algorithms, cited by 74.2%. Additional benefits highlighted by students included faster and improved learning processes, more accessible and comprehensible explanations and solutions compared to traditional teacher-provided explanations, as well as increased self-confidence and enhanced motivation to learn.

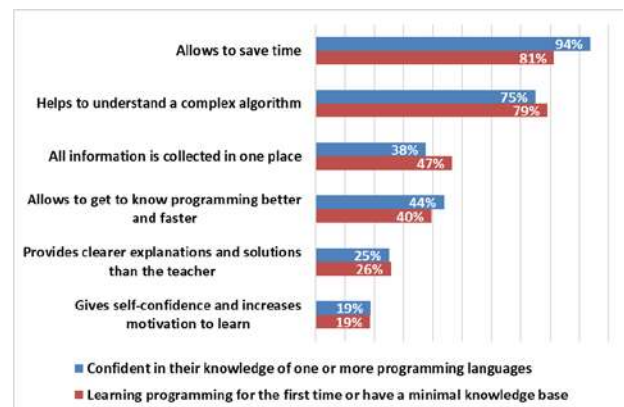


Figure 3 What advantages can you identify from using generative AI?

While a small percentage of students (2.9%) did not perceive any disadvantages in using generative AI for programming tasks, the majority identified several notable negative aspects. Entry-level students with minimal programming knowledge most frequently pointed out the inaccuracy of the generated code (72%) and the risk of over-reliance on AI (44%), which they believed weakened their algorithmic thinking abilities (35%). Students with more advanced knowledge also highlighted frequent inaccuracies in the generated solutions (88%) and the issue of AI misinterpreting their requests, which often led to misleading results (50%). Other disadvantages noted by respondents included a perceived lack of creativity and variability in problem-solving.

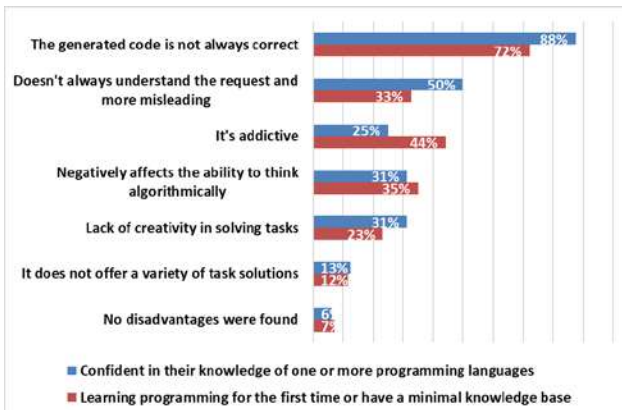


Figure 4 What disadvantages can you identify from using generative AI?

The survey results suggest that the majority of students (61%) require multiple clarifying queries to obtain a correct solution from generative AI, although 34% of respondents reported successfully resolving the task on the first attempt. These findings may indicate that many students have not yet developed the skills to accurately formulate queries to AI and tend to simply copy the task description and submit it without modification.

Furthermore, the survey revealed that students rarely present AI-generated solutions as their own work (only 11%). Instead, they use AI primarily to improve their own solutions: 43% of respondents integrate part of the generated code into their projects, and 46% modify the proposed code before submitting it for evaluation.

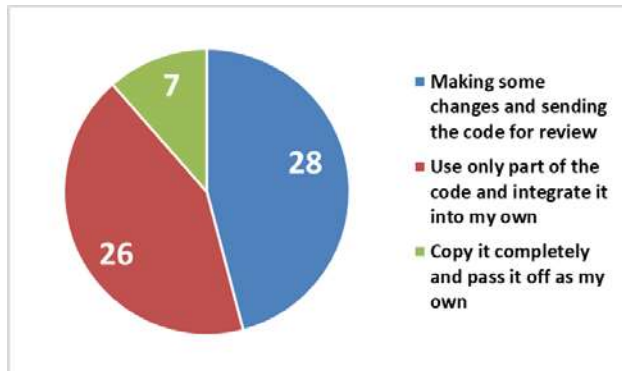


Figure 5 How do you use the AI-generated code?

The questionnaire also aimed to identify which generative AI models' students utilize, including ChatGPT, Gemini, YandexGPT, GigaChat, DeepSeek, Telegram-based chatbots, ChatInfo, and generative AI integrated into development environments. The obtained data served as the foundation for the second phase of the study.

One of the primary objectives of this study was to identify the differences in problem-solving approaches between generative AI models and students. While an experienced instructor can typically distinguish between independently written and externally generated code, the challenge lies in training an automated system to detect such instances and respond appropriately in real time.

Generative AI adheres strictly to the programming standards, ensuring consistent code formatting and comprehensive comments—an approach that is uncommon among most students. Furthermore, AI-generated solutions often incorporate advanced programming language features that are not typically covered at the initial stages of learning,

which may raise suspicions during the assessment of student work. Additionally, variations in variable naming conventions can serve as an indicator of AI involvement: while AI tends to assign names based on the semantic meaning of the variable's contents, students are more likely to use abstract or inconsistent naming patterns.

The identification of AI-generated code can be facilitated through various quantitative and qualitative metrics. Execution speed and memory consumption serve as fundamental indicators, as AI-generated code is often optimized for performance. This optimization stems from the efficient utilization of built-in programming language functions and advanced algorithms. In contrast, student-written code may exhibit higher memory consumption due to redundant variables or inefficient data structures.

Additional quantitative metrics indicative of GAI usage include features that are less commonly observed in student solutions, such as the frequency of inline comments, function definitions (including lambda functions), try-except blocks, and list comprehensions.

Furthermore, qualitative metrics can provide deeper insights into AI-generated code characteristics. These include adherence to programming style standards, the average length of variable and function names, Halstead Difficulty, which quantifies cognitive complexity in code comprehension, and Cyclomatic Complexity, a metric that assesses algorithmic complexity by measuring the number of linearly independent execution paths. The integration of these metrics enables a more comprehensive analysis of code provenance and aids in distinguishing between human-written and AI-generated solutions.

#### 4 CONCLUSION

The findings of this study provide an opportunity to evaluate the feasibility of integrating generative AI into the educational process. On the one hand, AI can serve as a tool for personalized student support, adapting to their level of proficiency and individual learning needs. On the other hand, its widespread use may adversely affect the development of independent problem-solving skills and critical thinking. These results highlight the necessity for further research on AI integration in education, with a particular focus on developing methodologies that foster precise problem formulation skills and promote the effective use of AI as an educational aid without diminishing students' cognitive engagement and motivation for independent work.

The analysis conducted in this research can serve as a basis for the development and integration of advanced software modules designed to detect AI-generated code or establish new principles of formulating the problems conditions in order to limit the extent of using GAI by the student while solving the problems. One potential implementation is a neural network embedded within the LMS, which would enable the automated identification of both AI-generated code and instances of code reuse among students. Furthermore, such a system could offer personalized recommendations for code optimization and refactoring, while also facilitating the implementation of individualized learning trajectories, thereby enhancing the adaptability and overall efficacy of programming education.

## 5 REFERENCES

- [1] See <https://disk.yandex.ru/d/b45m7TMG49mSUg>
- [2] Yilmaz R., Karaoglan Yilmaz F.G. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. *Computers in Human Behaviour: Artificial Humans*, 1(2).
- [3] Khaniev R.M. (2024). Use of innovative technologies in the learning process in higher education. *Problems and prospects of development of social economic and humanitarian sciences: pedagogy, psychology, economics, jurisprudence*, 119-124.
- [4] Valiakhmetova N.R., Akhmadullina R.M., Yarmakeev I.E. (2024). Opportunities and risks of applying neural networks in education. *Philology and Culture*, 2, 260-271.
- [5] Karmanova E.V. (2021). Automated control in teaching Python programming using CodeRunner LMS MOODLE plugin. *Science, informatisation, technologies, education*, 102-108.

### Contact information:

**Egor CHEGLOV**, senior lecturer, Doctoral student  
Novosibirsk State University of Economics and Management  
Kamenskaya st., 52, Novosibirsk, Novosibirsk region, Russian Federation, 630099  
[e.r.cheglov@edu.nsuem.ru](mailto:e.r.cheglov@edu.nsuem.ru)

**Andrei PESTUNOV**, PhD in Physics and Mathematics, Associate Professor  
Novosibirsk State University of Economics and Management  
Kamenskaya st., 52, Novosibirsk, Novosibirsk region, Russian Federation, 630099  
[a.i.pestunov@nsuem.ru](mailto:a.i.pestunov@nsuem.ru)

**Olga SHVETS**, PhD in Engineering Science, Associate Professor  
Novosibirsk State University of Economics and Management  
Kamenskaya st., 52, Novosibirsk, Novosibirsk region, Russian Federation, 630099  
[o.y.shvec@edu.nsuem.ru](mailto:o.y.shvec@edu.nsuem.ru)  
<https://orcid.org/0000-0001-5710-9056>