

# Improving Methodology for Developing Programming Competencies of Higher Educational Institutions' Students

Utepbergenova Aysuliu Tilepbergenovna

Trainee Teacher of Nukus State Pedagogical Institute, Uzbekistan

**Received:** 26 July 2026; **Accepted:** 23 August 2026; **Published:** 18 September 2026

**Abstract:** In today's digital economy, programming has become one of the most fundamental competencies for students in higher education institutions. As technological progress accelerates, industries increasingly demand graduates who possess not only theoretical knowledge but also practical programming skills, problem-solving abilities, and adaptability. Nevertheless, many students struggle with mastering programming due to its abstract nature and complex logic. Consequently, the effectiveness of educational methodology plays a decisive role in shaping students' professional competencies. This article provides an extended analysis of the current challenges in teaching programming, explores international practices, and proposes methodological improvements such as project-based learning, gamification, and the integration of digital tools. The discussion emphasizes that methodological innovation is not a supplementary element but a necessary condition for preparing competitive graduates.

**Keywords:** Programming competencies, higher education, methodology, coding skills, digital economy, student-centered learning.

**Introduction:** It is widely recognized that programming skills have become indispensable in the 21st century. Since almost every industry—ranging from finance and healthcare to engineering and education—relies on software, universities must ensure that students acquire solid programming competencies. However, although the importance of programming is clear, the pathway to effective teaching and learning is often complicated.

On the one hand, traditional methods such as lectures and textbooks remain useful for conveying fundamental concepts. On the other hand, they are insufficient when it comes to developing applied coding skills. As a result, many scholars have emphasized the importance of methodological innovation. For instance, Wing introduced the concept of computational thinking, arguing that programming should not be treated merely as a technical subject but rather as a universal problem-solving skill applicable across disciplines [5, 33-35]. Similarly, Kiesler stresses that competency-based

learning frameworks are more effective than content-driven approaches, since they focus on developing practical skills rather than memorizing syntax [2].

Furthermore, Vrabec, N., and Furtáková, L. highlight that project-based learning significantly enhances student engagement, because it links theory with real-world applications [4, 5-17]. Their research demonstrates that students who participate in project-oriented courses show higher retention rates and deeper understanding of complex programming concepts. In addition, Adizova notes that the digital economy requires higher education institutions to foster lifelong learning attitudes, meaning that programming education should prepare students to continuously update their skills [1, 126-131].

Therefore, the improvement of methodology in programming education is not merely a matter of academic interest. Rather, it is a critical requirement for national development, technological innovation, and economic growth. By linking theory with practice,

encouraging creativity, and promoting self-directed learning, universities can significantly enhance students' programming competencies.

Programming competencies extend far beyond the ability to type code in a particular language. In fact, they include several interrelated skills. Firstly, students must develop algorithmic thinking, which involves breaking down problems into logical steps. Secondly, they need analytical abilities, enabling them to evaluate different solutions. Thirdly, they must acquire debugging skills, since real-world coding always involves errors and corrections. Finally, programming also requires soft skills, including teamwork, communication, and adaptability, because most software development takes place in collaborative environments.

Thus, programming education in higher education institutions should not be viewed narrowly as technical training. Instead, it should be considered a holistic process that develops both cognitive and professional competencies.

Although programming is central to modern education, students often face serious difficulties. To begin with, many find abstract concepts—such as recursion, object-oriented programming, or data structures—extremely challenging. Moreover, since programming requires continuous practice, students who lack motivation quickly fall behind.

In addition, higher education institutions themselves face several obstacles. For example, some universities lack qualified instructors who are equally skilled in pedagogy and programming. Furthermore, curricula often remain outdated, focusing excessively on syntax rather than problem-solving. Likewise, assessment systems frequently emphasize written examinations rather than hands-on coding ability.

Moreover, there is a psychological barrier: programming is often perceived as “too difficult,” which discourages beginners. As a result, drop-out rates in introductory programming courses are significantly higher than in many other subjects. Therefore, overcoming these barriers requires methodological innovation, motivational strategies, and continuous institutional support.

In order to improve methodology, it is useful to examine international experiences. For instance, in Finland, programming is introduced to students as early as

primary school through interactive games and simple coding platforms. This early exposure reduces anxiety and builds confidence.

Similarly, Singapore integrates programming education with interdisciplinary courses, ensuring that students understand coding not as an isolated skill but as a tool for problem-solving in different fields. Moreover, South Korea emphasizes close collaboration between universities and industry partners, allowing students to work on real-world projects that reflect current market demands.

These international examples suggest that effective programming education combines early exposure, interdisciplinary integration, and practical relevance. Consequently, higher education institutions in other countries should adapt such practices to their own cultural and institutional contexts.

Given the challenges and international practices, several methodological improvements can be suggested.

First and foremost, programming courses should shift from teacher-centered lectures to student-centered approaches. By allowing students to explore problems independently and collaboratively, they develop a deeper understanding of programming concepts.

Furthermore, PBL has proven to be highly effective in programming education. Instead of solving isolated exercises, students work on projects that integrate different programming concepts. As a result, they not only gain technical knowledge but also develop teamwork, communication, and time management skills.

In addition, modern platforms such as GitHub, online compilers, coding simulators, and learning management systems should be integrated into programming education. These tools provide instant feedback, encourage collaboration, and expose students to real-world development environments.

Moreover, gamification strategies—such as coding challenges, digital badges, and leaderboards—can significantly increase motivation. Because students are rewarded for progress, they are more likely to engage in continuous practice, which is essential for mastering programming.

Finally, assessment should go beyond traditional

examinations. Instead, continuous assessment based on projects, portfolios, peer evaluation, and coding competitions can provide a more accurate reflection of students' programming competencies.

The improvement of methodology in programming education has broader implications. Firstly, it prepares graduates who are competitive in the global labor market. Secondly, it fosters innovation, since students learn to approach problems creatively. Thirdly, it strengthens national digital infrastructure by producing skilled specialists who can contribute to technological development.

Furthermore, methodological innovation promotes inclusivity. For instance, gamified learning and digital tools can make programming more accessible to students who initially lack confidence. As a result, a wider range of students—including those from non-technical backgrounds—can successfully acquire programming competencies.

## **CONCLUSION**

To sum up, improving the methodology of programming education in higher education institutions is an urgent necessity in the digital age. Although students face multiple challenges, these can be addressed through student-centered learning, project-based approaches, integration of technology, gamification, and continuous assessment. Moreover, international experiences demonstrate that early exposure, interdisciplinary learning, and collaboration with industry partners significantly enhance outcomes.

Therefore, universities must adopt a holistic strategy that not only imparts technical knowledge but also cultivates problem-solving, creativity, and collaboration. In this way, programming education will not only meet academic requirements but also serve as a foundation for national innovation, economic competitiveness, and individual professional success.

## **REFERENCES**

1. Adizova, M. (2025). DEVELOPING EFFECTIVE METHODOLOGIES FOR IMPROVING STUDENT COMPETENCE IN PROGRAMMING TECHNOLOGY EDUCATION. *Journal of Applied Science and Social Science*, 1(3), 126-131.
2. Kiesler, N. Modeling Programming Competency.
3. van der Vlies, R. (2020). Digital strategies in

education across OECD countries: Exploring education policies on digital technologies. *OECD Education Working Papers*, (226), 0\_1-45.

4. Vrabec, N., & Furtáková, L. (2024). Ways of defining digital competences and their components in the EU, EC and UNESCO recommendations. *Annales Universitatis Paedagogicae Cracoviensis. Studia de Cultura*, 16(2), 5-17.
5. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.