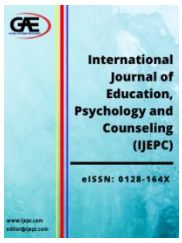
International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39



INTERNATIONAL JOURNAL OF EDUCATION, PSYCHOLOGY AND COUNSELLING (IJEPC)
https://gaexcellence.com/ijepc

PROGRAMMING EDUCATION IN HIGHER INSTITUTION: A CHRONOLOGICAL REVIEW

Nor Zalina Ismail[1], Mohd Rizal Razak[2]*, Saffa Raihan Zainal Abidin[3], Mohd Khairul Ikhwan Zolkefley[4], Mohamad Affendi Abdul Malek[5], Nor 'Aqilah Misman[6], Mohd Norafizal Abd Aziz[7]

[1]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Pahang Branch Raub Campus, Pahang, Malaysia
nza1601@uitm.edu.my https://orcid.org/0000-0001-7798-1941
[2]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Pahang Branch Raub Campus, Pahang, Malaysia
dragon_admire007@uitm.edu.my https://orcid.org/0000-0002-5052-756X
[3]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Pahang Branch Raub Campus, Pahang, Malaysia
saffaraihan@uitm.edu.my https://orcid.org/0000-0002-4735-6704
[4]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Pahang Branch Raub Campus, Pahang, Malaysia
khairulikhwan@uitm.edu.my https://orcid.org/0000-0001-6275-982X
[5]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Pahang Branch Raub Campus, Pahang, Malaysia
maffendi@uitm.edu.my https://orcid.org/0009-0001-3869-2546
[6]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Pahang Branch Raub Campus, Pahang, Malaysia
noraqilahmisman@uitm.edu.my https://orcid.org/0009-0007-2119-3517
[7]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Pahang Branch Raub Campus, Pahang, Malaysia
mnorafizal@uitm.edu.my https://orcid.org/0000-0002-4323-7193
*Corresponding Author

Article Info:

Article history:
Received date: 28.09.2026
Revised date: 13.10.2026
Accepted date: 02.11.2026
Published date: 01.12.2026

Abstract:

Computational thinking and coding skills are becoming a crucial part of higher education, and programming education is now a mandatory part of it. Nonetheless, the history of research on programming education has changed significantly over the years, and pedagogical strategies, learning technologies, assessment, and new educational issues have

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

To cite this document:

Ismail, N. Z., Razak, M. R., Abidin, S. R. Z., Zolkefly, M. K. I., Malek M. A. A., Misman, N. 'A., & Aziz, M. N. A. (2026). Programming Education in Higher Institution: A Chronological Review. *International Journal of Education, Psychology and Counselling, 11*(65), 20-39.

changed. Although the literature is increasing, there is a dearth of a chronological knowledge of how programming education research has been born, grown, and developed in higher education. The purpose of this study is thus to critically review the history of research in the field of programming education in higher education by critically reviewing the literature chronologically. The Scopus database was searched systematically with the main keywords being programming education and higher institution to find relevant scholarly publications. After the predetermined search and selection, 88 primary studies (n = 88) were included in the analysis and grouped into three chronological periods: (1) Emergence and Foundational Development (19982014), (2) Expansion and Consolidation (20152023), and (3) Accelerated Growth and Research Maturation (20242026). The chronological review shows that research interest has evolved over time and that there has been a shift in the focus of research to more varied, technology-enhanced, learner-centred, and research-based methods. The results give a systematic overview of the evolving research environment and emphasize the way programming education has evolved to meet pedagogical, technological, and learner-based challenges in higher education. The research adds a time dimension that can guide researchers and educators to determine the already set directions, new priorities and future research opportunities in programming education.

DOI: 10.35631/IJEPC.1165002

Keyword:

Higher Institution, Programming Education

© The authors (2026). This is an Open Access article distributed under the terms of the Creative Commons Attribution (CC BY NC) (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact ijepc@gaexcllence.com.

Introduction

Higher-level programming education is central to modern higher education since programming is no longer solely a technical skill in computer science, but a generalizable problem-solving approach applicable in STEM and STEAM fields (Malmi et al., 2025; Pan and Harun, 2025; Vinueza-Morales, Rodas-Silva, and Vidal-Silva, 2025). Programming has become an important graduate skill in universities, extending beyond traditionally technical majors, as it has been found to teach algorithmic thinking, abstraction, logical reasoning, creativity, and the capability to automate solutions to complex problems (Vinueza-Morales, Rodas-Silva, and Vidal-Silva, 2025). The process of learning to program in higher education is thus much more than just memorizing syntax; it entails students to analyze, break down problems, model data and processes, and refine solutions through the process of design-oriented thinking (Romero et al., 2017; Rovshenov and Sarsar, 2023; Vinueza-Morales, Rodas-Silva, and Vidal-Silva, 2025) .

The study of computational thinking supports this perspective by making programming one of the primary pathways through which students can be taught to become effective problem solvers in an increasingly digitalized workplace (Agbo et al., 2019; Rovshenov and Sarsar,

2023; Tikva and Tambouris, 2021). Meanwhile, the body of research on programming education has expanded significantly, both in terms of publication activity over the last few years and in terms of the number of studies, which is indicative of the global nature of the subject and its central role in computing education research (Malmi et al., 2025; Rolland Sobral, 2021; Rovshenov and Sarsar, 2023; Vinueza-Morales, Rodas-Silva, and Vidal-Silva, 2025). However, this growth has also shown that programming education is a developing discipline that still requires more synthesis, more frameworks, and more equitable application in institutional and national settings (Moosaei and Molinero, 2025; Vinueza-Morales, Rodas-Silva, and Vidal-Silva, 2025). Since digital technologies are transforming higher education, programming education should also keep up with the evolving industry needs, new tools, and diverse student needs to ensure that its educational value in the long term can be maintained (Vinueza-Morales, Rodas-Silva, and Vidal-Silva, 2025).

Although important, programming education in higher education is generally considered to be difficult both to students and teachers, particularly in introductory courses where failure and dropout rates are high (Medeiros et al., 2019; Moosaei and Molinero, 2025; Pan and Harun, 2025; Rolland Sobral, 2021; Rovshenov and Sarsar, 2023). Problem solving, mathematical preparedness, motivation, syntax, and abstract reasoning to convert ideas into executable programs are the challenges that students may face (Medeiros et al., 2019; Pan and Harun, 2025; Rovshenov and Sarsar, 2023). These cognitive requirements are further complicated by the lack of equal exposure to programming in the past, variations in the level of soft and hard skills among students, and psychological obstacles such as anxiety and fear, which may decrease engagement, performance, and persistence (Moosaei and Molinero, 2025; Pan and Harun, 2025) Teaching issues are also critical since instructors have to work with large and heterogeneous groups of students, and they often lack adaptive enough approaches, individualized support systems, or a thorough pedagogical model that would be applicable in the context of programming courses (Medeiros et al., 2019; Moosaei and Molinero, 2025; Xia, 2017) . Reviews and bibliometric studies have provided evidence that traditional teacher-centered methods are frequently ineffective, and student-centered, collaborative, and practice-oriented methods are more likely to facilitate programming learning (Moosaei and Molinero, 2025; Rolland Sobral, 2021; Vinueza-Morales, Rodas-Silva, and Vidal-Silva, 2025). The collaboration, contextualization of knowledge, anxiety reduction, and enhancement of higher-order thinking have been improved through approaches like project-based learning, pair programming, flipped classes, team-based learning, creative programming, and course design based on computational thinking (Agbo et al., 2019; Pan and Harun, 2025; Romero et al., 2017; Vinueza-Morales, Rodas-Silva, Vidal-Silva, et al., 2025).

This landscape is further complicated by new technological conditions: web-based and personalized environments have opened up possibilities of flexibility, access to resources, and distance learning, but also raise quality-management issues and require a careful integration of pedagogy (Xia, 2017). Infrastructure constraints still define the possible in resource-constrained contexts, and mobile compilers and online console tools are of interest as practical aids to programming education (Eteng et al., 2022). The latest development is the use of AI tools, including ChatGPT and code generators, as both valuable aids and a significant pedagogical threat: they can give hints, suggest code, and provide feedback quickly, yet they also lead to the development of dependency and the loss of independent conceptual knowledge when used without critical thinking (Becker et al., 2022; Da Silva et al., 2024). Therefore, the main challenge of higher institutions is not to teach students to code, but to create inclusive, evidence-based, and flexible programming education spaces that can foster conceptual

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

knowledge, independence, teamwork, and resilience in a fast-paced digital environment (Pan & Harun, 2025; Vinueza-Morales, Rodas-Silva, Vidal-Silva, et al., 2025).

Materials and Methods

Data Collection

Computational thinking, software development, and digital competencies have become essential elements of higher education, with programming education becoming an integral part of academic disciplines and professional practice. Universities around the world have thus increased programming courses in computer science, information technology, engineering and interdisciplinary programmes and researchers have investigated various pedagogical strategies to enhance the level of programming knowledge, problem solving skills, engagement and academic performance of students. In spite of this increasing body of scholarship, the study of programming education has changed significantly over the years, due to the shifts in educational technologies, teaching methods, programming paradigms, evaluation procedures, and the needs of learners. Prior research was mainly based on traditional methods of instruction and how to develop basic programming skills, but later research began to pay more attention to active and collaborative learning, automated evaluation, visualization tools, game-based learning, flipped classes, online and blended classes, and more recently, new technologies such as artificial intelligence. This evolution is significant to comprehend since programming education does not evolve in a linear fashion; instead, it is influenced by the evolving educational priorities, technological possibilities, and problems faced by higher education institutions.

A chronological overview thus offers a valuable lens in determining the changes in research priorities over various times, what educational issues have remained constant, and what new trends could shape future programming education. To create a solid evidence base on such an analysis, this study uses a systematic single step data collection procedure through the Scopus database, which is chosen due to its multidisciplinary coverage, standard indexing criteria, and comprehensive bibliographic data. The identification of relevant publications was performed with the help of a predefined search strategy with the help of carefully chosen keywords associated with the subject of programming education in higher education, and the records retrieved were analyzed in relation to the relevance and inclusion criteria of the study to create the final dataset. Limiting the source of data and using a uniform selection process improves the transparency, reproducibility, and methodological consistency of the review. Strict and methodical data gathering is especially important in chronological studies since discrepancies in database coverage, search terms, selection of publications, or screening processes may skew the apparent publication trends and give rise to false interpretations of time-related alterations. Well-designed data, in turn, offers a more reliable basis to study the changes over time, define the time frame of rising or falling, and explain the changes in the interests of research. Through a systematic review of the academic history of programming education in higher education, this review aims to offer an evidence-based narrative of the evolution of the field and to explain its key research directions, well-established themes, and new directions.

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

Table 1: The Specification for Primary Data Searching.

Keyword	**Programming Education**
Inclusion	Article, Journal, English, computer science
Exclusion	Besides computer science, review articles, other languages
Final Search String (Scopus)	TITLE-ABS-KEY (("programming education" OR "programming instruction" OR "computer programming education" OR "software development education" OR "coding education" OR "coding instruction") AND ("higher education" OR "higher institution" OR university OR college OR "tertiary education")) AND (LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (SRCTYPE , "j")) AND (LIMIT-TO (PUBSTAGE , "final")) AND (LIMIT-TO (EXACTKEYWORD , "Programming Education")) AND (LIMIT-TO (SUBJAREA , "COMP"))
Number of Primary Article	88

The data collection plan was aimed at developing a narrow and replicable evidence base to study the evolution of programming education in higher education. Table 1 shows the detail of the specification or primary data searching. The Scopus database was searched in a single-step search in the TITLE-ABS-KEY field to retrieve publications where programming education and higher education were directly mentioned in the title, abstract, or keywords. A broad range of related terms was used in the search strategy, which included: programming education, programming instruction, computer programming education, software development education, coding education, and coding instruction, with terms that reflect the higher education setting, such as higher education, university, college, and tertiary education. The combination was to enhance retrieval sensitivity and conceptual compatibility with the scope of the review. To enhance the quality and comparability of the evidence, the search was limited to the English language journal articles, final stages of publication, and the Computer Science subject area. Moreover, the keyword Programming Education was also used as a relevance criterion. Articles that were not reviews, were not in the field of Computer Science, or were in a non-English language were filtered out. These limitations are methodologically defensible since the research seeks to investigate the academic evolution of programming education in a well-defined disciplinary and educational framework as opposed to encompassing the wider literature on coding or computing education. As a result, the last search strategy yielded 88 primary articles, which offered a sufficiently narrowed corpus to be further chronologically analyzed.

The corpus of 88 articles obtained as a result gives a systematic basis on the interpretation of the way research on programming education in higher education has evolved throughout the years. Explicit inclusion and exclusion criteria are especially crucial in chronological review since time trends may be significantly affected by changes in search coverage and publication properties. The dataset might include heterogeneous types of documents, disciplines, languages, or secondary studies without systematic filtering, which would introduce noise and obscure actual changes in research activity. The limitation to peer-reviewed journal articles in Computer Science and the final publication stage increase the reliability of the records and minimizes the chances that the initial or non-comparable publications will bias the observed trend. Likewise, the use of the specific key word Programming Education enhances topical

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

specificity since the studies that are selected have a direct conceptual relationship with the theme of the review. However, the comparatively narrow dataset can be viewed as a reflection of the Scopus-covered Computer Science journal literature on programming education, and not of the overall body of research on programming education in the world. This difference matters to the validity and interpretability of later chronological discoveries, especially in evaluating times of greater publication, changes in research focus, or new themes. On the whole, the systematic selection of 88 primary articles creates a clear and reproducible data set on the basis of which the development of the research on programming education can be investigated with increased methodological confidence.

Data Clustering

Clustering of the publication data according to the temporal periods is a significant method of analysis in a chronological review since it allows the development of research activity to be viewed beyond the annual variations. Within the framework of Programming Education in Higher Institutions, as reflected in Figure 2, temporal clustering in terms of publication year and publication volume offers a methodical way of establishing different phases in the evolution, growth, and deepening of scholarly research in the field. The bibliographic information was obtained in Scopus, a well-known and extensive multi-disciplinary database, with the help of predefined keywords and advanced search techniques to guarantee the relevance and coverage of the recognized literature. By clustering the publication years based on the observable trends in research productivity, one can consider the periods of time when the same tendencies in the research productivity are observed, and, thus, such periods can be analyzed as a single entity, eliminating the possible distortion of the results by the single increase or decrease in the annual productivity. More to the point, this clustering increases the interpretability of longitudinal patterns of publication by showing the changes in the focus of research, new stages of the rapid growth of the field, and changes in the general state of the field. In this regard, the temporal clusters identified can offer a systematic point of departure to analyzing the way in which programming education research has changed over the years and can help to make better sense of the shifting research environment in higher education.

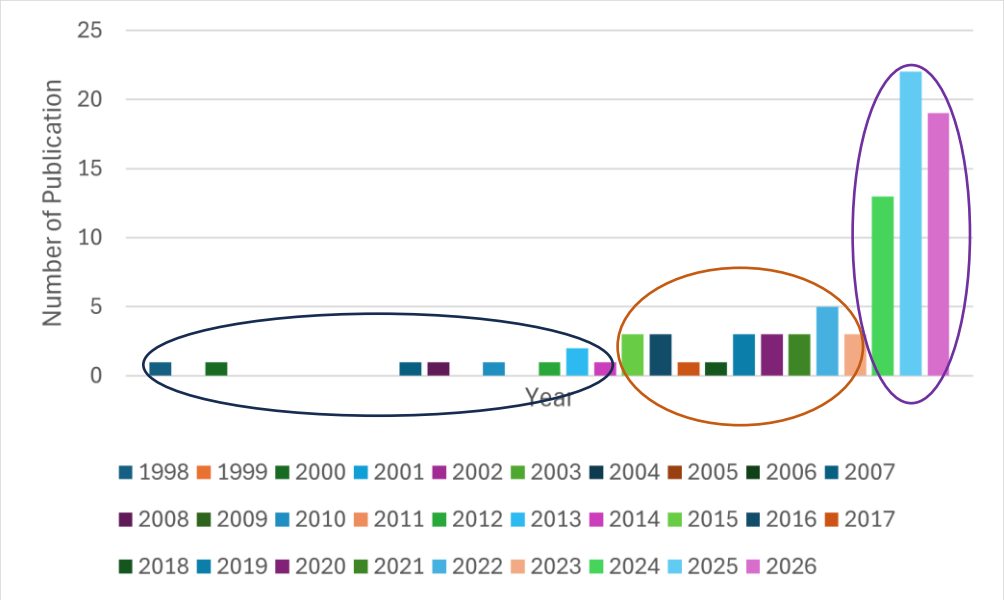


Figure 2: Number of Publication by Year

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

Phase 1: Emergence and Foundational Development (1998 – 2014)

The initial stage, Emergence and Foundational Development (1998 2014) is the oldest phase of the research on programming education in higher institutions. Publication activity in this period was very minimal and sparse with only 10 publications in 17 years. There were a few years where there were no publications, and the majority of active years had only one publication. The slight rise to two publications in 2013 was not yet a sign of a long-term growth. This trend indicates that programming education was not yet an extremely visible or actively explored research field in the literature that is indexed by Scopus. Studies of this foundational era can thus be viewed as initial exploration, where researchers started to investigate problems related to the teaching of programming, learning challenges, teaching methods, and how students can develop their programming skills.

Phase 2: Expansion and Consolidation (2015 – 2023)

The second stage, Expansion and Consolidation (2015 -2023) is characterized by a shift in the intermittent research activity to a more consistent publication pattern. It started with three publications in 2015 and 2016, then there were some fluctuations in 2017 -2021 with one to three publications. Nonetheless, the number of publications has since grown to five in 2022, followed by a slight decrease to three in 2023. In the nine years, 24 publications were created, which is more than the total of the 1998 -2014 phase. This growth suggests that programming education was getting wider and more enduring academic interest. The convergence can be attributed to the increasing awareness of the ongoing issues related to programming learning, as well as the rising exploration of teaching methods, learning technologies, assessment methods, and student-centred pedagogies in higher education. However, the comparatively average yearly production indicates that the field was still in its early stages of development and not in its significant growth.

Phase 3: Accelerated Growth and Research Maturation (2024 – 2026)

The third stage, Accelerated Growth and Research Maturation (20242026) is a significant shift in the research environment. The number of publications rose dramatically with 13 publications in 2024, 22 publications in 2025 and 19 publications in 2026. Though 2026 is the index of publications up to August, it has already surpassed the annual rates of the previous period, which lasted nearly the whole period before it. This phase alone has 54 publications, which is about 61.4 percent of the total publications in the dataset. The dramatic growth means that the programming education has entered the era of high research productivity and growing scholarly visibility. This acceleration can be linked to the development of new pedagogical and technological issues, such as artificial intelligence, automated programming support, online and blended learning, new programming pedagogies, and evolving expectations of computational competencies. This stage, therefore, can be seen not only as quantitative expansion, but as a sign of the maturation and diversification of programming education research in higher education.

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

Result and Discussion

Emergence and Foundational Development (1998 – 2014)

The Emergence and Foundational Development period (1998 2014) can be further subdivided into three significant periods according to the evolving focus of research in programming education: 1998 2000, 2007 2010, and 2012 2014. In 1998 2000, the focus of research was primarily on the underlying problems of programming-language selection and program comprehension. In a comparison between Java and Modula-3, Böszoermenyi (1998) emphasized the pedagogical significance of the choice of the first programming language, especially in terms of supporting the understanding and mastering of the basic concepts. Ueno (2000) later applied this issue to intelligent programming support by creating ALPUS II, which proved the possibility of generalized program understanding between Pascal and C. In 2007 - 2010, the emphasis was on technology-assisted and learner-centered teaching. Nakajima et al. (2007) created an adaptive question-answer system that paired exercises with the level of skills of the learners and assisted in the practice of iterative programming, whereas Kalles (2008) focused on code inspection, teamwork, implementation skills and preparation of larger programming projects using a university programming laboratory. Koelling (2010) also illustrated the possibilities of Greenfoot to facilitate learning of programming using graphical and interactive Java-based programs. Taken together, these studies suggest that there is an early shift in the choice of appropriate programming foundations to more interactive, adaptive, collaborative and technology-supported learning environments.

The 2012 -2014 era showed an additional expansion of research interests to particular learning and teaching issues in programming education. The algorithmic visualization component of the TSUNA-TASTE environment was developed by Teramoto et al. (2012), and the results showed that the visual representation might be of some use to beginners studying C. In 2013, the research was expanded to academic integrity and conceptual understanding. Chunhui et al. (2013) dealt with programming-code plagiarism by preventive measures and a Java-based detection system, noting that it is challenging to detect suspicious code manually in a large number of programming tasks. Similarly, Cetin (2013) discovered that the students of the university had significant problems with the comprehension of classes, objects, and the relationships between them in object-oriented programming. These challenges were linked to the findings that overreliance on code-writing activities, examples, and improperly constructed analogies were connected to these difficulties, and visualization was found to be an effective way to enhance conceptual knowledge. These papers suggest that the research in programming education was growing more worried about not just instructional technology but also understanding algorithms, academic dishonesty, and conceptual challenges faced by students.

Brito and De Sá-Soares (2014) reinforced the emphasis on assessment and early detection of programming-learning challenges, stating that students enrolled in introductory programming courses might have a problem with identifying the distinction between learning programming concepts and applying them to problem-solving. The results were in favor of higher assessment frequency as a tool of offering timely student progress indicators and being able to take corrective action before the challenges escalated. Throughout the 1998 -2014 period, the results thus indicate a progressive growth of the underlying programming-language and program-understanding problems, to adaptive and interactive learning systems, and then to visualization, conceptual knowledge, plagiarism detection, and formative assessment. Despite the comparatively low level of publication activity at this time, the research papers set a number

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39
of key research directions that define the nature of programming education in higher educational institutions, including the use of technology to aid learning, the construction of conceptual knowledge in students, ongoing assessment, and how to cope with difficulties that come with programming coursework.

Expansion and Consolidation (2015 – 2023)

The Expansion and Consolidation phase is characterized by a vivid expansion of the scope of programming education beyond the traditional classroom-based teaching to technology-enhanced, interactive and learner-centred methods. Initial research focused on expanding the reach of programming education to online and practical education. As Kurhila and Vihavainen (2015) have shown, an MOOC at the university level could not only expand the reach of CS education to Finnish school students but also serve as an alternative route to university education. In the same way, Takioka et al. (2015) demonstrated that the enhancement of teaching strategies and the focus on practice helped students to create programming applications more frequently. Engagement and motivation were further enhanced by the introduction of visualization, gamification, game-based learning, and accessible programming environments (Kölling, 2016; Mozelius et al., 2016; Olsson et al., 2015). Technology-supported course redesign has also yielded positive results in higher education, such as a decrease in dropout rates and signs of better grades, especially when active learning, collaboration, and automated assessment were incorporated into programming courses (Kaila et al., 2016).

Between 2017 and 2021, programming education started to emphasize more on scalability, automated feedback, collaborative learning, and student engagement. The Murphy et al. (2017) UK-wide survey found that Java was still popular in introductory university programming even though respondents thought Python was easier to teach and learn, indicating the ongoing controversy over the choice of introductory programming languages. It was also during this time that web-based programming environments and automated assessment systems emerged that minimized technical barriers and offered immediate feedback to learners (Hameer and Pientka, 2019; Yazaki et al., 2018) Programming education was also linked to more broad developmental outcomes, with intrinsic motivation being found to be a significant factor in positive youth development through programming (Kong and Wang, 2019), and visual tools like turtle graphics were created to make early Java learning more intuitive and motivating (Mackin, 2019). Technology-mediated and collaborative methods later evolved to problem-based groupware, online programming courses, online judges, and robotics-based systems. According to these studies, there is a gradual transition between merely providing programming content and developing interactive environments that facilitate collaboration, practical experience, feedback, and learner motivation (Bilegjargal and Hsueh, 2021; Byun et al., 2021; Chorfi et al., 2011; Lee and Yi, 2021).

By 20222023, the field of programming education research was more mature, with more data-driven, personalized, intelligent, and AI-assisted research. A study of online programming education has shown that behavioural engagement, especially video viewing and self-assessment, correlates with academic performance, and learning-behaviour data may be analysed to determine student groups and predict performance at an early stage (Hsueh et al., 2022; Mai et al., 2022). There was also an increase in automated and intelligent technologies to enhance assessment and instructional support, such as code-quality assessment to identify individual contributions in programming projects, recommendation systems to provide similar

programming problems, and visualization tools to help instructors understand the process of source-code development by students (Chen et al., 2022; Taniguchi et al., 2022; Yoshimura et al., 2022). By the conclusion of the phase, generative AI became a new facet of programming education, and Yilmaz and Karaoglan Yilmaz (2023) found that students using ChatGPT had better computational thinking, programming self-efficacy, and motivation. However, there were still issues with programming problems, teacher readiness, pedagogy, and technological change (Gökoğlu and Kilic, 2023). The evolution of intelligent tutoring systems also reflected the increased focus on personalized learning and the integration of skills, and Huang et al. (2023) found that code tracing achieved significant learning outcomes when it was supported by instructional targets. Taken together, these papers suggest that the 20152023 decade is a significant synthesis of research in programming education, moving beyond access and engagement to adaptive, evidence-based, and intelligent learning environments.

Accelerated Growth and Research Maturation (2024 – 2026)

Accelerated Growth and Research Maturation phase (2024 2026) indicates a significant diversification of programming education research, with 2024 research being heavily oriented towards enhancing learning effectiveness, learner support, and technology-enhanced pedagogy. Huang et al. (2024) found a composition effect that persisted in novice code tracing, where students had problems when multiple programming skills needed to be combined, and conceptual misunderstanding and errors in the process due to cognitive load were found to be significant. The use of technology-based practices gained momentum in ChatGPT feedback, gamification, flipped learning, serious games, online learning, visualization, and interactive problem-solving systems. Ouyang et al. (2024) discovered that ChatGPT and instructor feedback yielded similar programming knowledge and product quality, but each facilitated different types of collaborative discourse. Mellado et al. (2024) found that gamification had various learning outcomes based on gender, and Malkoc et al. (2024) found that academic self-concept, learning engagement, course satisfaction, and achievement goals were all relevant in online flipped programming courses. Hainey and Baxter (2024) showed that serious games are acceptable as a programming revision and assessment method, but Oka et al. (2024) found that there is a positive correlation between repeated use of a guided problem-solving system and performance on programming tasks. Other papers touched on online participation (Ochi and Tateno, 2024), live coding based on various learning preferences (Masegosa et al. (2024), predicting programming aptitude using machine learning (Nakada and Miura, 2024), and flipped learning with the help of learning analytics (Iio, 2024). Meanwhile, the quick development of generative AI brought new educational and ethical issues. Groothuijsen et al. (2024) discovered that ChatGPT assisted in debugging, conceptual knowledge, and code generation but might adversely impact quality of code and pair programming, whereas Humble et al. (2024) discovered learning opportunities and risks associated with cheating, unreliable responses, and assessment. In general, the year 2024 suggests a shift to individualized, interactive, data-driven, and AI-enhanced programming education, without losing the focus on cognitive development, learner differences, academic integrity, and meaningful human interaction.

In 2025, the scope of research shifted to the larger-scale integration, assessment, and management of artificial intelligence and personalized learning in programming education. According to Cheung et al. (2025), videoconferencing-based online programming education was moderately effective at the time of the COVID-19, and Python, C++, and Java were the most commonly taught languages. Lepp and Kaimre (2025) discovered that AI chatbots were

often used by students to complete programming tasks, yet increased reported use was negatively correlated with academic performance, highlighting the need to take proper guidance. A number of studies investigated GenAI as a learning and instructional aid: Cubillos et al. (2025) did not find a significant difference in learning between GenAI and video-based instruction, but each method yielded different motivational results; Kohen-Vacs et al. (2025) reported positive perceptions of GenAI but increased difficulty when students had to correct the code generated by the LLM; and Tseng et al. (2025) demonstrated that an AI-based code-review system could potentially reduce the cognitive-load increment caused by peer review. Qu et al. (2025) also found that few-shot prompt-based evaluation led to better learning outcomes, code quality, and assessment efficiency. Vinueza-Morales, Rodas-Silva, Vidal-Silva, et al. (2025) found project-based learning, flipped classes, and collaborative programming to be significant teaching methods, with the newer technologies of augmented and virtual reality. The topics of personalization, diversity of learners, self-regulation, and inclusion also gained more and more research. Boguslawski et al. (2025) discovered that LLMs could enhance learner autonomy and competence, but not social support, whereas Marco-Galindo et al. (2025) reported that individualized feedback and structured interaction enhanced participation among students re-taking introductory programming classes, but did not increase overall success. Wang et al. (2025) distinguished four groups of learners based on the patterns of GPT-use, whereas Oyelere and Aruleba (2025) emphasized the differences in the attitude towards AI-supported programming education among countries and educational levels. By 2025, thus, the literature had outgrown merely testing educational technologies to analyzing how AI, personalization, automation of assessment, characteristics of learners, ethical regulation, and inclusive access ought to be incorporated into programming education. This wider trend is supported by Malmi et al. (2025); Zhu and Zhang (2025), and others, which highlight the growing size, variety, and the interdisciplinary nature of computing and programming education research.

The 2026 literature shows further maturation of research with more advanced AI applications, learning analytics, automated evaluation, adaptive systems, self-regulated learning, and process-oriented analysis, and ongoing programming-learning challenges are still evident. Sarmento and Reis (2026) found source-code comments as possible scaffolds to comprehension, debugging, reflection, and formative assessment, but Pešić et al. (2026) demonstrated that LLMs could assist with automated generation and analysis of algorithm-complexity tasks but was not yet reliable enough to be fully automated. Bikanga Ada (2026) affirmed that the programming challenges are linked to conceptual knowledge, learning environments, and learning resources even with the existence of ChatGPT. Strickroth and Azaiz (2026) discovered that peer code review on large scale can assist in detecting errors and providing coding guidance, but students tend to overlook minor errors. Gao et al. (2026) and Bouallegue et al. (2026) reported more advanced adaptive methods, finding three relatively stable patterns of programming-learning in the longitudinal analysis of Gao et al., and high predictive accuracy of an XGBoost model in identifying students who were at risk of poor performance. The use of AI in pedagogy also became more organized: Abdullah and Al Musawi (2026) discovered that a guided GenAI approach was significantly better at code analysis, debugging, optimization, and problem-solving than unconstrained use of GenAI; (Jang and Oh (2026) found that combined flipped learning and GenAI led to better computational thinking and problem-solving; and Alanazi et al. (2026) found that the successful implementation of AI required pedagogical appropriateness, institutional preparedness, infrastructure, policy, and faculty involvement. Other papers described the benefits of visual scripting (Villagómez-Palacios et al., 2026), emotion-sensitive chatbot engagement (Nawahdah et al., 2026),

programming robots (Fuse & Noguchi, 2026), automated awareness of GenAI abuse (Karnalim, 2026), and educational games (Tassuov et al., 2026). Meanwhile, the literature on AI-related research points to the necessity to strike a balance between technological support and autonomous thinking, creativity, teamwork, academic honesty, and fundamental programming knowledge. In this way, the 20242026 phase may be defined as a transition to evidenced-based and responsible technology integration, where the research in programming education focuses more on not only whether an intervention works, but also on who it works, under what conditions, with what cognitive and educational outcomes, and how such technologies can be integrated into higher education in a sustainable way.

Conclusions

This chronological review explored the development of research in the field of programming education in higher institutions between 1998 and 2026 through a systematic search of the Scopus database, leading to a narrow corpus of 88 primary articles. The review divided the literature into three stages of development: Emergence and Foundational Development (1998 -2014), Expansion and Consolidation (2015 -2023), and Accelerated Growth and Research Maturation (2024 -2026). This chronological arrangement offers a systematic foundation of the study of how the education of programming has evolved in line with evolving pedagogical requirements, technological strengths, student traits, and educational issues. The initial stage was marked by small and occasional publication efforts, and research focused on selection of programming languages, understanding of programs, program visualization, program evaluation, and learning challenges at the foundations. The latter stage revealed more extensive and enduring academic interest, and the research gradually shifted to the focus on learner-centred, collaborative, technology-enhanced, online, gamified, and automated learning conditions. The third stage was a significant acceleration with 54 publications or a percentage of 61.4 of the entire data set and showed the maturity and diversity of the field.

Throughout the history, one may see a distinct shift in the traditional programming education to adaptability, interactivity, personalization, data-driven, and smart educational settings. The development of methods The methodological development of research has moved towards more or less narrow studies of instructional practices and program challenges with empirical evaluation, automated assessment, learning analytics, predictive modelling and more complex adaptive systems. Specifically, the advent of generative AI has placed a significant new dimension in the research of programming education. Recent research is beginning to focus more on AI-assisted coding, debugging, code evaluation, personalized learning, academic integrity, learner autonomy, and what it takes to adopt AI responsibly. The data also reveal that technological progress does not eradicate chronic educational issues, especially the lack of conceptual understanding, critical thinking, diversity of learners, and the necessity of meaningful human communication. Therefore, the chronological development is not only a quantitative development, but a shift of inquiry how programming can be taught to a discussion of how programming can be taught effectively, adaptively, ethically and responsibly in technology-rich contexts.

The chronological structure makes a contribution to knowledge by showing the developmental associations that might be less observable in the traditional thematic reviews. The clustering of studies on the basis of the time of publication allows differentiating between the underlying issues and subsequent unifications as well as new research priorities, and minimizes the impact of the specific annual effects on the interpretation. This view also has practical implications to

higher institutions as the observed progression indicates that the key response to effective pedagogical practices should be matched with the use of suitable technologies instead of technological adoption being seen as a solution in itself. Future studies are thus recommended to explore sustainable and evidence-based infrastructure of AI, learning analytics, adaptive systems, collaborative learning, and personalized support without losing conceptual knowledge, creativity, autonomous problem-solving, and academic integrity. The use of interdisciplinary research and real-world educational environments should also be given more consideration to identify how emerging approaches will work in various institutional contexts, with various populations of learners, with various programming languages, and under various resource conditions. The current review has limitations that include use of one database, search terms and scope of disciplinary focus, use of English language Computer Science journal articles, and lack of full indexing of 2026, which is currently searching articles up to August. These limitations may be overcome in future chronological reviews by including several bibliographic databases, more interdisciplinary terminology, longer observation times, and cross-educational comparisons. In general, chronological analysis is an effective way to capture the dynamic nature of the research on programming education and to understand how the established practices transform into new paradigms and how it can serve as the basis of temporal evidence to inform future innovation, pedagogical decision-making, and evidence-based development in higher education.

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

Acknowledgements: The authors would like to sincerely thank Universiti Teknologi MARA (UiTM) Cawangan Pahang, Kampus Raub, for giving them the chance and assistance to conduct research thru the authorized leave to attend the research course. In order to improve research expertise, create the report, and finish the current study, this support offered crucial time and a suitable setting. The institution's dedication to fostering academic growth and research output, which greatly aided in the successful completion of this study and the publication that resulted, is greatly appreciated by the authors.

Funding Statement: No Funding

Conflict of Interest Statement: Regarding the publishing of this work, the authors state that they have no conflicts of interest. Each author has contributed to this work and given their approval for the manuscript's final submission to the journal.

Ethics Statement: There were no animals, human subjects, or sensitive data in this study that needed ethical clearance. The authors attest that the study was carried out in compliance with recognized guidelines for ethical publishing and academic integrity.

Author Contribution Statement: Each author made a substantial contribution to the manuscript's development. The study's conception, methodology, and general oversight were handled by Mohd Rizal Razak and Nor Zalina Ismail. Data collection, analysis, and result interpretation were handled by Saffa Raihan Zainal Abidin, Mohd Khairul Ikhwan Zolkefley and Mohamad Affendi Abdul Malek. Nor 'Aqilah Misman and Mohd Norafizal Abd Aziz helped with the manuscript's critical revision, drafting, and literature review. Before the manuscript was submitted, all authors read it and gave their approval.

International Journal of Education, Psychology and Counseling
IJEPC
EISSN : 0128-164X
Volume 11 Issue 65 (December 2026) PP. 20-39

References

Abdullah, S., & Al Musawi, A. S. (2026). Balancing innovation and integrity in using generative AI for a new computer programming assignments approach. *Journal of Education and E-Learning Research*, *13*(1), 140–149. https://doi.org/10.20448/jeelr.v13i1.8231

Agbo, J., Oyelere, S., Suhonen, J., & Adewumi, S. (2019). A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. https://doi.org/10.1145/3364510.3364521

Alanazi, M., Li, A., Almalawi, A., Samra, H., & Soh, B. (2026). Pedagogical Transformation and Teaching Practice in Programming Education Through AI Coding Assistants: Faculty Perspectives and the AI Coding Assistant Adoption Framework. *Applied Sciences (Switzerland)*, *16*(10). https://doi.org/10.3390/app16104833

Becker, B., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. (2022). Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. https://doi.org/10.1145/3545945.3569759

Bikanga Ada, M. (2026). Programming Challenges and Perceptions: A Study of Separate Groups before and after the Release of ChatGPT. *ACM Transactions on Computing Education*, *26*(2). https://doi.org/10.1145/3777904

Bilegjargal, D., & Hsueh, N.-L. (2021). Understanding students' acceptance of online judge system in programming courses: A structural equation modeling approach. *IEEE Access*, *9*, 152606–152615. https://doi.org/10.1109/ACCESS.2021.3126896

Boguslawski, S., Deer, R., & Dawson, M. G. (2025). Programming education and learner motivation in the age of generative AI: student and educator perspectives. *Information and Learning Science*, *126*(1–2), 91–109. https://doi.org/10.1108/ILS-10-2023-0163

Böszörményi, L. (1998). Why Java is not my favorite first-course language. *Software-Concepts and Tools*, *19*(3), 141–145. https://doi.org/10.1007/s003780050017

Bouallegue, S., Omri, A., & Al-Naemi, S. (2026). Machine Learning Approaches for Early Student Performance Prediction in Programming Education. *Information (Switzerland)*, *17*(1). https://doi.org/10.3390/info17010060

Brito, M. A., & De Sá-Soares, F. (2014). Assessment frequency in introductory computer programming disciplines. *Computers in Human Behavior*, *30*, 623–628. https://doi.org/10.1016/j.chb.2013.07.044

Byun, J., Park, J., & Oh, A. (2021). Cocode: Providing Social Presence with Co-learner Screen Sharing in Online Programming Classes. *Proceedings of the ACM on Human-Computer Interaction*, *5*(CSCW2). https://doi.org/10.1145/3476041

Cetin, I. (2013). Visualization: A tool for enhancing students' concept images of basic object-oriented concepts. *Computer Science Education*, *23*(1), 1–23. https://doi.org/10.1080/08993408.2012.760903

Chen, H.-M., Nguyen, B.-A., & Dow, C.-R. (2022). Code-quality evaluation scheme for assessment of student contributions to programming projects. *Journal of Systems and Software*, *188*. https://doi.org/10.1016/j.jss.2022.111273

Cheung, C. Y., Chiu, L. M., Su, Y., Chiu, D. K. W., & Ho, K. K. W. (2025). Effectiveness of Online Higher Education Programming Education During COVID-19 Under the Lens on the 5E Instructional Model. *IT Professional*, *27*(2), 75–83. https://doi.org/10.1109/MITP.2024.3466512

Chorfi, A., Hedjazi, D., Aouag, S., & Boubiche, D. (2011). Problem-based collaborative learning groupware to improve computer programming skills. *Behaviour and Information Technology*, *0*(0), 1–20. https://doi.org/10.1080/0144929X.2020.1795263

Chunhui, W., Zhiguo, L., & Dongsheng, L. (2013). Preventing and detecting plagiarism in programming course. *International Journal of Security and Its Applications*, *7*(5), 269–278. https://doi.org/10.14257/ijsia.2013.7.5.25

Cubillos, C., Mellado, R., Cabrera-Paniagua, D., & Urra, E. (2025). Generative Artificial Intelligence in Computer Programming: Does It Enhance Learning, Motivation, and the Learning Environment? *IEEE Access*, *13*, 40438–40455. https://doi.org/10.1109/ACCESS.2025.3532883

Da Silva, C. A. G., Ramos, F., De Moraes, R., & Santos, E. L. Dos. (2024). ChatGPT: Challenges and Benefits in Software Programming for Higher Education. *Sustainability*. https://doi.org/10.3390/su16031245

Eteng, I., Akpotuzor, S., Akinola, S., & Agbonlahor, I. (2022). A Review on Effective Approach to Teaching Computer Programming to Undergraduates in Developing Countries. *Scientific African*. https://doi.org/10.1016/j.sciaf.2022.e01240

Fuse, I., & Noguchi, T. (2026). Effectiveness of Tool Use in Robot-Based Programming Education for Courses with Learners of Varying Knowledge Levels. *Journal of Robotics and Mechatronics*, *38*(1), 276–287. https://doi.org/10.20965/jrm.2026.p0276

Gao, Z., Yan, H., Huang, Y., Zhang, X., Saqr, M., Sun, X., & Feng, J. (2026). A complex system approach to decode different learning patterns in programming between majors: score, engagement, and problem-solving efficiency. *Smart Learning Environments*, *13*(1). https://doi.org/10.1186/s40561-026-00431-7

Gökoğlu, S., & Kilic, S. (2023). Programming learning and teaching of pre-service computer science teachers: Challenges, concerns, and solutions. *E-Learning and Digital Media*, *20*(5), 498–518. https://doi.org/10.1177/20427530221117331

Groothuijsen, S., van den Beemt, A., Remmers, J. C., & van Meeuwen, L. W. (2024). AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. *Computers and Education: Artificial Intelligence*, *7*. https://doi.org/10.1016/j.caeai.2024.100290

Hainey, T., & Baxter, G. (2024). A Serious game for programming in higher education. *Computers and Education: X Reality*, *4*. https://doi.org/10.1016/j.cexr.2024.100061

Hameer, A., & Pientka, B. (2019). Teaching the art of functional programming using automated grading (experience report). *Proceedings of the ACM on Programming Languages*, *3*(ICFP). https://doi.org/10.1145/3341719

Hsueh, N.-L., Daramsenge, B., & Lai, L.-C. (2022). EXPLORING THE INFLUENCE OF STUDENTS' MODES OF BEHAVIORAL ENGAGEMENT IN AN ONLINE PROGRAMMING COURSE USING THE PARTIAL LEAST SQUARES STRUCTURAL EQUATION MODELING APPROACH. *Journal of Information Technology Education: Research*, *21*, 403–423. https://doi.org/10.28945/5010

Huang, Y., Brusilovsky, P., Guerra, J., Koedinger, K., & Schunn, C. (2023). Supporting skill integration in an intelligent tutoring system for code tracing. *Journal of Computer Assisted Learning*, *39*(2), 477–500. https://doi.org/10.1111/jcal.12757

Huang, Y., Schunn, C. D., Guerra, J., & Brusilovsky, P. (2024). Why Students Cannot Easily Integrate Component Skills: An Investigation of the Composition Effect in Programming. *ACM Transactions on Computing Education*, *24*(3). https://doi.org/10.1145/3673239

Humble, N., Boustedt, J., Holmgren, H., Milutinovic, G., Seipel, S., & Östberg, A.-S. (2024). Cheaters or AI-Enhanced Learners: Consequences of ChatGPT for Programming

Education. *Electronic Journal of E-Learning*, *22*(2 Special Issue), 16–29. https://doi.org/10.34190/ejel.21.5.3154

Iio, J. (2024). An Evaluation of the Flipped Classroom Approach Toward Programming Education. *Journal of Information Processing*, *32*, 159–165. https://doi.org/10.2197/ipsjjip.32.159

Jang, E.-S., & Oh, K.-S. (2026). Investigation of the Influence of Flipped Learning and Text-Based Generative Artificial Intelligence in Programming Education: Effectiveness and Pedagogical Strategies. *Computer Applications in Engineering Education*, *34*(1). https://doi.org/10.1002/cae.70146

Kaila, E., Kurvinen, E., Lokkila, E., & Laakso, M.-J. (2016). Redesigning an object-oriented programming course. *ACM Transactions on Computing Education*, *16*(4). https://doi.org/10.1145/2906362

Kalles, D. (2008). Students working for students on programming courses. *Computers and Education*, *50*(1), 91–97. https://doi.org/10.1016/j.compedu.2006.03.003

Karnalim, O. (2026). Online Platform to Inform GenAI Misuse in Programming Through Code Differences. *International Journal of Distance Education Technologies*, *24*(1). https://doi.org/10.4018/IJDET.397045

Kohen-Vacs, D., Usher, M., & Jansen, M. (2025). Integrating Generative AI into Programming Education: Student Perceptions and the Challenge of Correcting AI Errors. *International Journal of Artificial Intelligence in Education*. https://doi.org/10.1007/s40593-025-00496-4

Kölling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education*, *10*(4). https://doi.org/10.1145/1868358.1868361

Kölling, M. (2016). Educational programming on the Raspberry Pi. *Electronics (Switzerland)*, *5*(3). https://doi.org/10.3390/electronics5030033

Kong, S.-C., & Wang, Y.-Q. (2019). Positive youth development from a "3Cs" programming perspective: a multi-study investigation in the university. *Computer Science Education*, *29*(4), 335–356. https://doi.org/10.1080/08993408.2019.1599646

Kurhila, J., & Vihavainen, A. (2015). A purposeful MOOC to alleviate insufficient cs education in Finnish schools. *ACM Transactions on Computing Education*, *15*(2). https://doi.org/10.1145/2716314

Lee, H.-J., & Yi, H. (2021). Development of an onboard robotic platform for embedded programming education. *Sensors*, *21*(11). https://doi.org/10.3390/s21113916

Lepp, M., & Kaimre, J. (2025). Does generative AI help in learning programming: Students' perceptions, reported use and relation to performance. *Computers in Human Behavior Reports*, *18*. https://doi.org/10.1016/j.chbr.2025.100642

Mackin, K. J. (2019). Turtle graphics for early Java programming education. *Artificial Life and Robotics*, *24*(3), 345–351. https://doi.org/10.1007/s10015-019-00528-y

Mai, T. T., Bezbradica, M., & Crane, M. (2022). Learning behaviours data in programming education: Community analysis and outcome prediction with cleaned data. *Future Generation Computer Systems*, *127*, 42–55. https://doi.org/10.1016/j.future.2021.08.026

Malkoc, S., Steinmaurer, A., Gütl, C., Luttenberger, S., & Paechter, M. (2024). Coding Decoded: Exploring Course Achievement and Gender Disparities in an Online Flipped Classroom Programming Course. *Education Sciences*, *14*(6). https://doi.org/10.3390/educsci14060634

Malmi, L., López-Pernas, S., Nguyen, H., Apiola, M., & Saqr, M. (2025). Topical Analysis of Programming Education Research Literature. *ACM Transactions on Computing Education*, *26*, 1–47. https://doi.org/10.1145/3767734

Marco-Galindo, M.-J., Minguillón, J., García-Solórzano, D., & Sancho-Vinuesa, T. (2025). Intervention for Students Repeating a CS1 Online Course: Analyzing Their Performance, Behavior, and Perceptions. *ACM Transactions on Computing Education*, *25*(3). https://doi.org/10.1145/3743685

Masegosa, A. R., Cabañas, R., Maldonado, A. D., & Morales, M. (2024). Learning Styles Impact Students' Perceptions on Active Learning Methodologies: A Case Study on the Use of Live Coding and Short Programming Exercises. *Education Sciences*, *14*(3). https://doi.org/10.3390/educsci14030250

Medeiros, R. P., Ramalho, G. L., & Falcao, T. P. (2019). A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Mellado, R., Cubillos, C., Vicari, R. M., & Gasca-Hurtado, G. (2024). Leveraging Gamification in ICT Education: Examining Gender Differences and Learning Outcomes in Programming Courses. *Applied Sciences (Switzerland)*, *14*(17). https://doi.org/10.3390/app14177933

Moosaei, R., & Molinero, G. (2025). Pedagogical Framework for Programming Courses in Higher Education. *2025 IEEE Global Engineering Education Conference (EDUCON)*, 1–5. https://doi.org/10.1109/educon62633.2025.11016289

Mozelius, P., Tomos, F., Shabalina, O., Miller, C., Malliarakis, C., Balan, O. C., & Chickerur, S. (2016). Game-based technologies in teaching programming in higher education: Theory and practices. *Recent Patents on Computer Science*, *9*(2), 105–113. https://doi.org/10.2174/2213275908666151030212745

Murphy, E., Crick, T., & Davenport, J. H. (2017). An Analysis of Introductory Programming Courses at UK Universities. *Art, Science, and Engineering of Programming*, *1*(2). https://doi.org/10.22152/programming-journal.org/2017/1/18

Nakada, T., & Miura, M. (2024). Extracting typing game keystroke patterns as potential indicators of programming aptitude. *Frontiers in Computer Science*, *6*. https://doi.org/10.3389/fcomp.2024.1412458

Nakajima, H., Takahashi, N., & Hosokawa, Y. (2007). A QA cycle for teaching programming. A mechanism for automatically posing questions corresponding to learner's skill. *Systems and Computers in Japan*, *38*(1), 28–40. https://doi.org/10.1002/scj.20471

Nawahdah, M., Sawalha, H., & Jabara, H. (2026). Enhancing programming education with emotion-aware chatbot interfaces: a Wizard-of-Oz study among Arabic-speaking university students. *International Journal of Educational Technology in Higher Education*, *23*(1). https://doi.org/10.1186/s41239-026-00600-1

Ochi, T., & Tateno, K. (2024). Comparing Efficacy of Online Real-time Classes with On-demand Viewing Classes. *Journal of Information Processing*, *32*, 133–138. https://doi.org/10.2197/ipsjjip.32.133

Oka, H., Ohnishi, A., Nishida, T., Terada, T., & Tsukamoto, M. (2024). A Choice-Based Programming Learning Method to Develop Problem-Solving Skills. *IEEE Access*, *12*, 119550–119562. https://doi.org/10.1109/ACCESS.2024.3443879

Olsson, M., Mozelius, P., & Collin, J. (2015). Visualisation and gamification of e-Learning and programming education. *Electronic Journal of E-Learning*, *13*(6), 441–454. https://www.scopus.com/pages/publications/84957593353?origin=resultslist

Ouyang, F., Guo, M., Zhang, N., Bai, X., & Jiao, P. (2024). Comparing the Effects of Instructor Manual Feedback and ChatGPT Intelligent Feedback on Collaborative Programming in China's Higher Education. *IEEE Transactions on Learning Technologies*, *17*, 2227–2239. https://doi.org/10.1109/TLT.2024.3486749

Oyelere, S. S., & Aruleba, K. (2025). A comparative study of student perceptions on generative AI in programming education across Sub-Saharan Africa. *Computers and Education Open*, *8*. https://doi.org/10.1016/j.caeo.2025.100245

Pan, Y., & Harun, J. (2025). Conquering Coding Fears: A Systematic Review of Programming Anxiety in Higher Education. *J. Inf. Technol. Educ. Res.*, *24*, 20. https://doi.org/10.28945/5543

Pešić, Đ., Vujošević Janičić, M., Mišić, M., & Protić, J. (2026). Assessing the effectiveness of large language models for generating and estimating time complexity of code segments. *PeerJ Computer Science*, *12*. https://doi.org/10.7717/peerj-cs.3946

Qu, Y., Li, F., Dou, X., Feng, C., & Li, B. (2025). Research on Object-Oriented Programming Teaching Evaluation Method Based on Few-Shot Prompt Learning. *Journal of Internet Technology*, *26*(7), 893–900. https://doi.org/10.70003/160792642025122607005

Rolland Sobral, S. (2021). *Strategies on Teaching Introducing to Programming in Higher Education*. 133–150. https://doi.org/10.1007/978-3-030-72660-7_14

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*(1). https://doi.org/10.1186/s41239-017-0080-z

Rovshenov, A., & Sarsar, F. (2023). Research Trends in Programming Education: A Systematic Review of the Articles Published Between 2012-2020. *Journal of Educational Technology and Online Learning*. https://doi.org/10.31681/jetol.1201010

Sarmento, G., & Reis, M. J. C. S. (2026). The Pedagogical Role of Source Code Comments in Programming Education: A Qualitative Systematic Review with Layered Evidence. *Informatics in Education*, *25*(2), 167–203. https://doi.org/10.15388/infedu.2603.031

Strickroth, S., & Azaiz, I. (2026). Qualitative analysis of peer reviews of a large introductory programming course. *Computer Science Education*, *36*(2), 354–374. https://doi.org/10.1080/08993408.2025.2450587

Takaoka, E., Yonezu, T., & Namekawa, T. (2015). Programming education at high schools and universities: Design, development, and assessment. *IPSJ Online Transactions*, *8*(2015), 1–8. https://doi.org/10.2197/ipsjtrans.8.1

Taniguchi, Y., Minematsu, T., Okubo, F., & Shimada, A. (2022). Visualizing Source-Code Evolution for Understanding Class-Wide Programming Processes. *Sustainability (Switzerland)*, *14*(13). https://doi.org/10.3390/su14138084

Tassuov, B., Nadyrbekova, A., Ibragimova, K., Taszhurekova, Z., Niyetbayeva, N., Abzhapparova, A., & Aidaraliyeva, Z. (2026). Educational games as a tool for teaching programming in digital extracurricular computer science education. *Frontiers in Computer Science*, *8*. https://doi.org/10.3389/fcomp.2026.1766830

Teramoto, T., Onishi, A., Koshii, S., & Kawata, S. (2012). A programming e-learning system in education-support PSE: TSUNA-TASTE. *Advances in Information Sciences and Service Sciences*, *4*(14), 33–45. https://doi.org/10.4156/AISS.vol4.issue14.5

Tikva, C., & Tambouris, E. (2021). A systematic mapping study on teaching and learning Computational Thinking through programming in higher education. *Thinking Skills and Creativity*, *41*. https://doi.org/10.1016/j.tsc.2021.100849

Tseng, C.-H., Lin, H.-C. K., Huang, A. C.-W., & Lin, J.-R. (2025). Investigating the effects of PuppyCodeReview, an AI-based code review system, on students' cognitive load. *SoftwareX*, *31*. https://doi.org/10.1016/j.softx.2025.102283

Ueno, H. (2000). A generalized knowledge-based approach to comprehend pascal and c programs. *IEICE Transactions on Information and Systems*, *E83-D*(4), 591–598. https://www.scopus.com/pages/publications/0033687336?origin=resultslist

Villagómez-Palacios, Á., Fuente-Burdiles, C. D. L., & Vidal-Silva, C. (2026). Videogame Programming & Education: Enhancing Programming Skills Through Unity Visual Scripting. *Computers*, *15*(1). https://doi.org/10.3390/computers15010068

Vinueza-Morales, M., Rodas-Silva, J., & Vidal-Silva, C. (2025). Teaching Programming in Higher Education: Analyzing Trends, Technologies, and Pedagogical Approaches Through a Bibliometric Lens. *International Journal of Advanced Computer Science and Applications*. https://doi.org/10.14569/ijacsa.2025.0160106

Vinueza-Morales, M., Rodas-Silva, J., Vidal-Silva, C., Córdova-Morán, J., & Cevallos-Ayón, E. (2025). Teaching programming in higher education: a bibliometric analysis of trends, technologies, and pedagogical approaches. *Frontiers in Education*. https://doi.org/10.3389/feduc.2025.1525917

Wang, D., Dong, X., Fang, Z., Wang, L. Y. K., & Jin, Z. (2025). AI in the Classroom: GPT Usage and Learner Typologies in Programming. *Education Sciences*, *15*(10). https://doi.org/10.3390/educsci15101353

Xia, B. S. (2017). An In-Depth Analysis of Teaching Themes and the Quality of Teaching in Higher Education: Evidence from the Programming Education Environments. *The International Journal of Teaching and Learning in Higher Education*, *29*, 245–254. https://consensus.app/papers/an-indepth-analysis-of-teaching-themes-and-the-quality-of-xia/e1f9eaec8cec5ccab4d02ab85540c981/

Yazaki, S., Tsuchiya, H., & Ishihata, H. (2018). Development of a web-based front-end environment to aid programming lectures on unix-like systems. *Journal of Information Processing*, *26*, 376–385. https://doi.org/10.2197/ipsjjip.26.376

Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence*, *4*. https://doi.org/10.1016/j.caeai.2023.100147

Yoshimura, R., Sakamoto, K., Washizaki, H., & Fukazawa, Y. (2022). WOJR: A Recommendation System for Providing Similar Problems to Programming Assignments. *Applied System Innovation*, *5*(3). https://doi.org/10.3390/asi5030053

Zhu, M., & Zhang, K. (2025). Artificial Intelligence for Computer Science Education in Higher Education: A Systematic Review of Empirical Research Published in 2003–2023. *Technology, Knowledge and Learning*, *30*(4), 2417–2441. https://doi.org/10.1007/s10758-025-09859-1