INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH IN PROGRESSIVE EDUCATION & DEVELOPMENT

OPEN ACCESS JOURNAL

Vol 14, Issue 01, (2025) E-ISSN: 2226-6348

Evaluating the Difficulties in Programming Learning: Insights from Polytechnic Students

Zuraihah Ngadengon[1,2], Tamil Selvan Subramaniam[1], Zurina Yasak[1], Noor Azmiza Ideris[3], Zuraidah Mohd Ramly[4]

[1]Universiti Tun Hussein Onn Malaysia, Malaysia, [2]Politeknik Muadzam Shah, Malaysia, [3]Politeknik Sultan Abdul Halim Mu'adzam Shah, Malaysia, [4]Politeknik Mersing, Malaysia

Corresponding Author Email: tselvan@uthm.edu.my

To Link this Article: http://dx.doi.org/10.6007/IJARPED/v14-i1/24518 DOI:10.6007/IJARPED/v14-i1/24518

Published Online: 07 February 2025

Abstract

This study explores the challenges and perspectives of students learning programming at a Malaysian polytechnic. The study seeks to analyze students's understanding of Problem Solving and Program Design (PSPD), the factors that contribute to poor performance in the course and the influence learning environments have on how well they perform. We surveyed 236 students to obtain their demographic data and knowledge, skills, and attitudes toward programming education. The above shows that control structure topics were identified as problematic among students. Moreover, the study identifies a few challenges, including designing algorithms, debugging, and understanding programming syntax. The results also indicate that students favor more hands-on, applications-oriented approaches to the learning process, such as group discussion, pair programming, and laboratory work, rather than didactic lecture-based approaches. Consequently, the results from the study further revealed that students' reactions to computational thinking modules are' positive, which allows students to enhance their problem-solving and program design abilities. Seeing these outcomes indicates that some teaching methods, such as stress learning in pairs through pair programming and incorporating computational thinking, should be encouraged to improve the outcome of programming education. The exploration presents evidence-based strategies that lecturers can embed in their teaching to help alleviate students' struggles and understanding of programming principles.

Keywords: Computational Thinking, Education, Pair Programming, Polytechnic, Problem Solving and Program Design

Introduction

Software and the tech sector have boomed, and that boom has created demand for coding skills, which are ubiquitous and require more than just memorizing syntax. Learning programming requires cultivating analytical thinking, algorithmic reasoning, and the ability to convert abstract ideas into working code. All around the world, the mastery of programming is becoming even more important, given that it is one of the most sought-after competencies in the contemporary labour market and will continue to be relevant in the future (Amnouychokanant et al., 2021a; Kim & Lee, 2016; Żyła et al., 2024). According to Dengler

and Matthes (2018), individuals with programming expertise can secure well-paying jobs and remain in high demand in the fast-changing digital economy. Programming is the methodical process of instructing a computer to perform specific tasks using a language the system can understand, often involving the manipulation of binary code. Hence, learning programming involves exposing students to languages like Java, Python, and C, where they create software programs (Ching et al., 2018).

Proficiency in programming constitutes an essential competency requisite for students pursuing disciplines within the Information Technology and Computer Science program (Ubaidullah et al., 2021). The fundamental aim of the programming course is to cultivate expertise and proficiency, with the overarching goal of producing adept graduates poised for success in the programming domain (Rahim et al., 2018). Other than that, effective resolution of programming problems necessitates a confluence of diverse skills. These skills include systematic thinking (Papadakis et al., 2016), algorithmic thinking (Angeli, 2022; Malik et al., 2021), problem-solving skills (Erol & Çırak, 2022; A. F. Lai & Yang, 2022; Topalli & Cagiltay, 2018), creativity (Laura-Ochoa & Bedregal-Alpaca, 2022; Weng et al., 2023), cognitive abilities, and the strategic utilization of problem-solving approaches (Mohd Yusoff et al., 2021).

Many students face difficulties and barriers during the early stages of learning programming, especially novices (Alshaye et al., 2018; Cheah, 2020; Husin et al., 2020; Mohd Rum & Zolkepli, 2018; Zheng et al., 2022). Programming courses often have high failure and student dropout rates (Chuang & Chang, 2024; Figueiredo & García-Peñalvo, 2024; Huang et al., 2024; Roque-Hernandez et al., 2021). A study by Bennedsen and Caspersen (2007) determine the average failure rate of students in programming courses. Results showed that 33% of students failed on average across 63 institutions (12.7%) worldwide. All over the world, around 650,000 students fail programming courses every year, and they believe that the number of students who enrol in programming courses amounts to more than two million every year.

Subsequently, Watson and Li (2014) expanded this investigation to 15 countries and 51 institutions. Their data when they made the publication indicated a slightly better state of affairs, with pass rates for introductory programming courses at 67.7% and fail charges at 32.3%. Even with this small drop, their study showed a significant obstacle that a third of students in these classes face: dropping out or withdrawing. Advancing this line of inquiry, Bennedsen and Caspersen (2019) administered an even larger survey of 161 institutions around the world, with a much lower average failure rate of 28% in programming courses. However, these studies highlight the challenges that are still present in computer programming education. To make this point even clearer, Topalli and Cagiltay (2018) noted failure rates exceeding 50%, while Cárdenas-Cobo et al. (2021) indicate that the pass rate in programming courses did not exceed 43%, which reflects the continuing challenges found in this area of study.

Problem-solving in programming entails an in-depth comprehension of the challenge at hand and devising a well-thought-out strategy prior to commencing the coding phase (Cheah, 2020; Rahim et al., 2018). A major issue of teaching Computer Science is that students often have poor problem-solving skills (Cárdenas-Cobo et al., 2021; Kadar et al., 2021; Mohd Noor et al., 2021; Piwek & Savage, 2019). Furthermore, students often encounter difficulties in

comprehending questions and internalizing core concepts, which impedes their capacity to formulate strategic methodologies and effectively tackle programming problems (Mohd Rum & Zolkepli, 2018). According to Hai Hom and Abdul Talib (2020), students struggled to comprehend problem-solving questions and were incapable of planning steps to solve programming problems. The students also had difficulties understanding basic programming concepts (Azhar & Adnan, 2022). When they have the problem-solving phases, day-to-day focus on steps that students simply know to convert problem statements to get onto a PC program without actually going through the activities that would area of the solution (Malik & Coldwell-Neilson, 2017).

The learning of programming was shown to have a negative impact on students that can lead to bad outcomes in terms of academic grades and future careers. Robins et al. (2003) insist that early programming challenges result in students leaving and not having a strong self-image, which is critical to their pursuing futures in the domain. Correspondingly, problems in comprehension of programming concepts may have an impact on the overall academic performance of the students thus giving rise to poor performance in assignments as well as exams (Lahtinen et al., 2005).

Polytechnics are key institutions in Technical and Vocational Education and Training (TVET) which are beginning to tackle these challenges with programs designed to teach programming. Polytechnics form a part of the factors for a highly skilled national workforce. Thus, polytechnics are tasked with a dual responsibility to produce academically and technically competent graduates who are more ethical and fully enabled to meet the challenges of the 21st-century workforce. Students enrolled in the Diploma in Information Technology (DIT) program at Polytechnics will be taught as a part of the Polytechnics in DIT program. They will take the first course available, Problem Solving and Program Design (PSPD), through the first semester. PSPD serves as a prerequisite for the more advanced programming courses that follow. Therefore, mastering programming fundamentals is essential; without it, students may struggle to succeed in more advanced programming courses. This strong foundation is crucial for their future learning and development.

Specifically, the study aims to achieve the following objectives: (i) evaluate students' comprehension of key topics in the PSPD course; (ii) analyze students' learning programming difficulties; (iii) examine contributing factors to unsatisfactory performance in programming courses; (iv) explore optimal learning settings that facilitate learning programming and (v) analyze students' perceptions of effective programming learning strategies. By identifying these challenges, polytechnics can better support students in overcoming obstacles and enhancing their programming proficiency. The findings will offer evidence-based recommendations for lecturers to improve teaching approaches and facilitate more effective learning experiences. Students can achieve success in their academic and professional careers.

Research Methodology

This study, conducted at a Malaysian polytechnic, is a descriptive study that aims to investigate students' viewpoints on learning challenges and discern their preferences for effective methods of learning programming. Therefore, a quantitative survey was used to obtain relevant data to answer the research questions presented in this study. This study uses

a questionnaire instrument adapted and modified from the questionnaire of Tan et al. (2009). Two experts specializing in programming education and one expert in instructional design validated this instrument. They have been teaching programming and instructional design for more than five years.

Table 1 provides an overview of the questionnaire, divided into four sections. Section A gathers basic student demographics, giving us a snapshot of the participants' backgrounds. Meanwhile, Section B looks at the experience in programming and knowledge of computing, helping us understand how familiar the respondents are with programming. Section C focuses on learning difficulties, factors for unsatisfactory performance, settings that facilitate learning, and effective strategies in programming education, aiming to identify the obstacles students face and the approaches that work best in helping them learn. In Section A, descriptive statistics present the respondents' demographic data, utilizing frequency and percentage to provide insights into their backgrounds. Consequently, students will respond based on the provided scale for sections B to C. The items in Section B use a 4-point Likert scale, which is 1=Never, 2=Weak, 3=Moderate, and 4=Strong. Meanwhile, the items in Section C use a 5-point Likert scale, which is 1 = Strongly Disagree, 2 = Disagree, 3 = Uncertain, 4 = Agree, and 5 = Strongly Agree.

Table 1
Questionnaire information

Section	Description
A	Student demographics
B	Students' comprehension of key topics in the Problem Solving and Program Design course
C	Difficulties in learning programming
	Factors to unsatisfactory performance in learning programming
	Learning settings that facilitate learning programming
	Perceptions of effective programming learning strategies

A pilot study was conducted on 35 respondents at a polytechnic. After the pilot study, the data was analyzed to determine the level of reliability of the questionnaire items using Cronbach's alpha. According to Creswell (2011), reliability means that the scores of the instruments used in the study are stable and consistent. Generally, reliability is represented with a numerical coefficient between 0.0 and 1.0, where higher values indicate higher reliability (Gay et al., 2011). The Cronbach's alpha value for the questionnaire is 0.858. This shows that the items in the questionnaire are highly reliable. This is because a Cronbach's alpha value of more than 0.7 indicates that the item has a high level of reliability (Farisiyah et al., 2025; Hair et al., 2009; Nunnally & Bernstein, 1994). Data from the survey was analyzed using Statistical Program for Social Science (SPSS) version 29.0 to obtain mean scores, percentages, and standard deviations. The researcher analyzed the mean score using the interpretation of mean scores proposed by Nunnally and Bernstein (1994). Table 2 presents the interpretation of the mean scores from Nunnally and Bernstein's (1994), which four levels of suggested interpretation mean score.

Table 2
Mean Score Interpretation

Mean Scale	Level
1.00 – 2.00	Low
2.01 – 3.00	Medium Low
3.01 – 4.00	Medium High
4.01 – 5.00	High

Results and Discussion

Student Demographics

Section A of the survey questionnaire focused on the demographic information of the respondents participating in this study. This study involved 236 respondents from the DIT program who are enrolled in the PSPD course. Table 3 presents a summary of the demographic characteristics of the respondents, including gender, ethnicity, and eligibility for polytechnic admission. According to the data in Table 3, there were 143 male students (60.6%) and 93 female students (39.4%). The ethnic breakdown revealed that 193 students identified as Malay (81.8%), 30 as Indian (12.7%), nine as Chinese (3.8%), and four as belonging to other ethnic groups (1.7%). Concerning eligibility for polytechnic admission, the results revealed that 220 respondents (93.2%) had the Sijil Pelajaran Malaysia (SPM) as their entry qualification. In comparison, 15 respondents (6.4%) had a Community College Certificate, and one respondent (0.4%) belonged to other categories. Regarding programming experience in section B, 89 students (37.7%) reported having programming experience. Meanwhile, 147 students (62.3%) had no programming experience.

Table 3
Student Demographics

Item	Category	Frequency	Percentage (%)
Gender	Male	143	60.6
	Female	93	39.4
Race	Malay	193	81.8
	Chinese	9	3.8
	Indian	30	12.7
	Others	4	1.7
Eligibility to enter the polytechnic	SPM	220	93.2
	Community College	15	6.4
	Others	1	0.4
Proficient in using computers	Yes	236	100.0
	No	0	0.0
Experience in programming	Yes	89	37.7
	No	147	62.3

Evaluate students' comprehension of key topics in the Problem Solving and Program Design course

Table 4 illustrates the students' comprehension of topics related to the PSPD course. The table is organized from lowest to highest mean values. The table displays students' average scores and standard deviations for the topics. Overall, the mean values for all categories range from 2.89 to 3.08, indicating a medium level of understanding. Control structures in problem-

Solving involving sequence, selection, and repetition yielded a mean value of 2.89 (SD=0.706). This implies that control structures, which include sequence, selection, and repetition topics, are complex for students to learn (Aris, 2015; Caceffo et al., 2016; Cheah, 2020; Ma et al., 2011; Robins et al., 2003; Xinogalos, 2014).

Table 4
Comprehension of topics in the Problem Solving and Program Design Course

Topics	Mean	SD	Interpretation
Control structures in problem-solving (sequence, selection, repetition)	2.89	0.706	Medium Low
Using operators in a program	2.97	0.687	Medium Low
Types and patterns in algorithms to solve a problem	2.99	0.720	Medium Low
Basic programming language	3.01	0.699	Medium High
Problem-solving concept	3.02	0.687	Medium High
Data and identifier	3.05	0.663	Medium High
Programming life cycle	3.07	0.726	Medium High
Fundamentals of programming languages	3.08	0.680	Medium High

SD standard deviation

Analyze Students' Learning Programming Difficulties

The results, displayed in Table 5, are organized from the highest to the lowest mean values. The mean values, ranging from 3.22 to 3.34, indicate a medium-high difficulty in grasping essential programming concepts. The findings indicate that the highest mean value, 3.34 (SD = 0.916), shows that students struggle most with designing programs to solve specific tasks. This suggests they have difficulty creating algorithms, recognizing problems, devising solutions, and implementing their ideas through coding. Additionally, students face challenges with debugging, which has a mean value of 3.31 (SD = 0.960), and understanding syntax, with a mean value of 3.30 (SD = 0.864). These skills are fundamental to programming. Hence, structured techniques such as pseudocode or flowcharts can help students visualize and plan their solutions before coding, enhancing their understanding and problem-solving skills (Abidin et al., 2018; Hooshyar et al., 2015; Zhang et al., 2023). Therefore, lecturers should emphasize a step-by-step approach to problem-solving, encouraging students to break down complex problems into smaller, more manageable parts.

Table 5
Difficulties in Learning Programming

Difficulty	Mean	SD	Interpretation
Designing a program to solve specific tasks	3.34	0.916	Medium High
Finding errors from own program	3.31	0.960	Medium High
Learning the programming language syntax	3.30	0.864	Medium High
Understanding basic concepts of programming structure	3.28	0.926	Medium High
Using a program development environment	3.22	0.857	Medium High

SD standard deviation

Examine Contributing Factors to Unsatisfactory Performance in Programming Courses

The data presented in Table 6 highlight several factors contributing to students' unsatisfactory performance in a programming course. The two highest-ranking factors, with mean values of 3.32 and 3.17, are the syllabus's emphasis on theory over practical application and the extensive number of topics covered each semester, classified as "Medium High." These findings suggest that the syllabus's structure and content may significantly impact students' challenges in the course. Other factors, with mean values ranging from 2.93 to 2.52 and categorized as "Medium Low," include the quality of lecturer presentations and attention, insufficient practical examples, and a lack of effective teaching methods. Additionally, issues such as low student motivation, malfunctioning lab computers, and an unconducive learning environment hinder performance, although to a lesser extent than syllabus-related issues. The relatively low mean scores for these factors indicate moderate concerns. Nevertheless, they still affect student outcomes in the programming course.

Table 6
Factors Contributing to Unsatisfactory Performance Programming Courses

Factors	Mean	SD	Interpretation
The syllabus emphasizes theory over practical application	3.32	0.930	Medium High
The syllabus for each semester covers too many topics	3.17	0.791	Medium High
Lecturers' presentations and attention to students need improvement	2.93	1.099	Medium Low
Few practical examples are provided	2.71	1.115	Medium Low
The teaching methods are not highly effective	2.63	1.004	Medium Low
Students show low motivation toward learning	2.57	1.072	Medium Low
The computers in the labs do not function properly	2.55	1.061	Medium Low
The learning environment is not conducive to studying	2.52	1.013	Medium Low

SD standard deviation

Explore Optimal Learning Settings that Facilitate Learning Programming

The data presented in Table 7 highlights the learning environments that students find most conducive to learning programming. Interactive discussions with lecturers scored the highest mean value, 4.23 (SD = 0.805), indicating a "High" level of effectiveness in facilitating learning. Practical work in laboratory sessions is closely followed, with a mean score of 4.19 (SD = 0.800), also rated "High." This suggests that students highly value hands-on practice. Group discussions with peers received a mean score of 4.14 (SD = 0.871), and pair programming scored 4.03 (SD = 0.894), both classified as "High," emphasizing the significance of collaborative learning approaches. Lecture-based learning has a slightly lower mean score of 3.97 (SD = 0.901), categorized as "Medium High," indicating moderate effectiveness in this context. Solo programming received the lowest mean score of 3.07 (SD = 1.213) and was rated "Medium High". These results suggest a clear preference among students for interactive and collaborative learning environments. Collaborative learning positively impacts student collaboration in groups and the acquisition of programming knowledge (Zhan et al., 2024). Furthermore, collaborative learning can improve students' academic achievement

(Pimpimool, 2024) and learning outcomes (X. Lai & Wong, 2022; Toukiloglou & Xinogalos, 2024) in learning programming.

Table 7
Optimal Learning Settings that Facilitate Learning Programming

Learning settings	Mean	SD	Interpretation
Interactive discussions with lecturers	4.23	0.805	High
Practical work in laboratory sessions	4.19	0.800	High
Group discussions with peers	4.14	0.871	High
Pair programming with a partner	4.03	0.894	High
Engaging in lecture-based learning	3.97	0.901	Medium High
Solo programming	3.07	1.213	Medium High

SD standard deviation

Assessing Students' Perspectives on the Effectiveness of Modules in Programming Education
The data presented in Table 8 highlights the student perceptions of effective programming learning strategies. The mean score of 3.93 for the learning module indicates an essential perspective on its usefulness in learning problem-solving learning and program design. Similarly, students favourably view the inclusion of computational thinking elements, such as decomposition, abstraction, generalization, algorithms, and evaluation, with a mean score of 3.89 (SD=0.809) for improving problem-solving abilities. In contrast, opinions on solo programming are more neutral, with a mean score of 3.16 (SD=1.098). This reflects more significant variability in student opinions; some find it valuable for independent work, while others perceive it as less beneficial. Finally, pair programming, with a mean score of 3.94 (SD=0.838), showed that students strongly agreed with learning with peers. This demonstrates a consistent viewpoint that pair programming enhances problem-solving through peer feedback. The analysis indicates that polytechnic students largely agree on the benefits of structured learning modules, elements of computational thinking, and pair programming in improving their problem-solving and program design skills. Correspondingly, these findings suggest that integrating computational thinking, pair programming, and structured learning modules may offer a well-rounded approach to accommodating diverse learning preferences in programming education.

Table 8
Perceptions of Effective Programming Learning Strategies

Perspectives	Mean	SD	Interpretation
Learning modules help in learning problem-solving and program design	3.93	0.801	Medium High
Elements of computational thinking, such as decomposition, abstraction, generalization, algorithms, and evaluation, aid in learning problem-solving and program design	3.89	0.809	Medium High
Solo programming contributes to learning problem-solving and program design	3.16	1.098	Medium High
Pair programming is seen as beneficial for learning problem-solving and program design	3.94	0.838	Medium High

SD standard deviation

According to Lye & Koh (2014), lecturers can use various approaches to teach programming, including pair and solo programming. In industry, the time allocated by a software developer or programmer to perform work is 30% programming time alone or solo, 50% time with a partner (pair programming), and 20% time with two or more partners (Nagappan et al., 2003). Current programming education emphasizes individual learning or solo programming (Garcia, 2021). Implementing pair programming can transfer knowledge between students, and scaffolding occurs through communication (Demir & Seferoglu, 2021). Both solo and pair programming approaches have positive effects on learning programming. Students report that the solo programming approach makes them more confident and better understand the computer programs they produce (Simon & Hanks, 2008). Furthermore, solo programming positively affects coding achievement (Demir & Seferoglu, 2021). The study by Beasley & Johnson (2022) reported that the average assignment and exam scores for students who used the pair programming approach obtained higher scores. Therefore, in an academic context, learning programming can be carried out solo or in pairs by adapting the approach implemented by the industry.

Problems in learning programming may be caused by disorganized thinking and a lack of computational thinking elements (Habib et al., 2021). Computational thinking is an essential 21st-century skill for computer scientists and all humans (Morris & Liu, 2020). According to Lee & Cho (2020), computational thinking, the core of computer science, is an essential thinking process for solving problems by describing them so they can be solved effectively. Computational thinking skills are cognitive skills that can be used in teaching and learning the programming process (Mohd Yusoff et al., 2020). Moreover, emphasis on computational thinking elements has a positive impact on student achievement in various fields, which are mathematics (Columba, 2020; Mohd Fadzil et al., 2022), chemistry (Chongo et al., 2021), science (Lapawi & Husnin, 2020) and programming (Amnouychokanant et al., 2021b; Hai Hom & Abdul Talib, 2020; Namli & Aybek, 2022; Ou Yang et al., 2023; Polat & Yilmaz, 2022). Then, educators need to consider using strategies for better learning programming to help minimize student difficulties (Sabarinath & Quek, 2020) and thus help improve student achievement. Teaching and learning programming can be improved by developing innovative teaching approaches (Kadar et al., 2022). Therefore, the programming teaching approach to integrate computational thinking into teaching and learning should be applied as effectively as possible.

Conclusion

The results of this study provide the current state of programming education among polytechnic students, revealing the challenges and methods that can be used in learning programming. The study revealed that students struggle to design programs for specific tasks, debug errors, and understand programming syntax. These challenges stem from a heavy emphasis on theoretical content, a dense syllabus, and limited opportunities for hands-on practice, which are essential elements for building strong programming skills. The dense syllabus, in particular, makes it difficult for students to fully grasp basic programming concepts, leaving them overwhelmed and underprepared. One of the most challenging topics is control structures, which include sequencing, selection, and iteration. Students find this topic particularly difficult because it requires logical reasoning and abstract thinking, but still underdeveloped skills. Additionally, many students have no prior experience with programming, making it more difficult for them to grasp basic concepts such as algorithmic thinking and program design. This lack of foundational knowledge dramatically hinders their

ability to effectively apply problem-solving techniques, leaving them unprepared to meet the demands of a programming course.

Based on these findings, several suggestions can be made to improve programming education. First, lecturers should focus on practical methods when learning programming by ensuring that students can apply theoretical concepts in real-world scenarios. The use of structured problem-solving approaches, such as pseudocode and flowcharts, should be emphasized so that students can break down complex programming tasks. In addition, the syllabus should be reviewed to ensure a balance between theoretical content and practical applications. In addition, reducing the topics covered each semester allows for deeper exploration of core concepts. Integrating computational thinking involving decomposition, abstraction, generalization, algorithms, and evaluation can improve students' problem-solving and program design skills. Furthermore, incorporating more collaborative learning strategies, such as pair programming and peer feedback, can also increase student engagement and understanding. Interactive discussions between students and lecturers should be emphasized because students like this learning method. These findings suggest that an approach that integrates computational thinking, pair programming, and structured learning modules can improve students' understanding of programming. A more conducive teaching approach at polytechnics can help students overcome challenges and achieve greater success in programming education.

References

Abidin, A. F. Z., Yaacob, M. R. Bin, Diah, M. A. I. B. M., Kadiran, K. A., Mustapa, R. F., Abdullah, M. Bin, Ismail, M. I., & Zaiton, S. N. A. H. (2018). E-FLOWCHART: An electronic educational quiz board that test student knowledge on C programming concept using flowchart command. *ARPN Journal of Engineering and Applied Sciences*, *13*(23), 9081–9085.

Alshaye, I. A., Jumaat, N. F., & Tasir, Z. (2018). Programming skills and the relation in Fostering Students' Higher Order Thinking. *Asian Social Science*, *14*(11), 76. https://doi.org/10.5539/ass.v14n11p76

Amnouychokanant, V., Boonlue, S., Chuathong, S., & Thamwipat, K. (2021a). A study of first-year students' attitudes toward programming in the innovation in educational technology course. *Education Research International*, *2021*, 1–10. https://doi.org/10.1155/2021/9105342

Amnouychokanant, V., Boonlue, S., Chuathong, S., & Thamwipat, K. (2021b). Online learning using block-based programming to foster computational thinking abilities during the COVID-19 pandemic. *International Journal of Emerging Technologies in Learning*, *16*(13), 227–247. https://doi.org/10.3991/ijet.v16i13.22591

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction*, *31*, 100329. https://doi.org/10.1016/j.ijcci.2021.100329

Aris, H. (2015). Improving students performance in introductory programming subject: A case study. *10th International Conference on Computer Science and Education, ICCSE 2015*, 657–662. https://doi.org/10.1109/ICCSE.2015.7250328

Azhar, N., & Adnan, N. H. (2022). Mengkaji kelemahan dan kekuatan dalam PdP pengaturcaraan C #: Satu kajian kes. *Jurnal Dunia Pendidikan*, *4*(2), 280–293.

Beasley, Z., & Johnson, A. (2022). The impact of remote pair programming in an upper-level

CS course. *Conference on Innovation and Technology in Computer Science Education*, *1*, 235–240. https://doi.org/10.1145/3502718.3524772

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, *39*(2), 32–36. https://doi.org/10.1145/1272848.1272879

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in Introductory Programming-12 years later. *ACM Inroads*, *10*(2), 30–36. https://doi.org/https://doi.org/10.1145/3324888

Caceffo, R., Wolfman, S., Booth, K. S., & Azevedo, R. (2016). Developing a computer science concept inventory for introductory programming. *SIGCSE 2016 - Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 364–369. https://doi.org/10.1145/2839509.2844559

Cárdenas-Cobo, J., Puris, A., Novoa-hernández, P., Parra-jiménez, Á., Moreno-león, J., & Benavides, D. (2021). Using scratch to improve learning programming in college students: A positive experience from a non-WEIRD country. *Electronics*, *10*, 1–15. https://doi.org/10.3390/electronics10101180

Cheah, C. S. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. *Contemporary Educational Technology*, *12*(2), ep272. https://doi.org/10.30935/cedtech/8247

Ching, Y. H., Hsu, Y. C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends*, *62*(6), 563–573. https://doi.org/10.1007/s11528-018-0292-7

Chongo, S., Osman, K., & Nayan, N. A. (2021). Impact of the plugged-in and unplugged chemistry computational thinking modules on achievement in chemistry. *Eurasia Journal of Mathematics, Science and Technology Education*, *17*(4), 1–21. https://doi.org/10.29333/ejmste/10789

Chuang, Y. T., & Chang, H. Y. (2024). Analyzing novice and competent programmers' problem-solving behaviors using an automated evaluation system. *Science of Computer Programming*, *237*, 103138. https://doi.org/10.1016/j.scico.2024.103138

Columba, L. (2020). Computational thinking using the first in math® online program. *Mathematics Teaching-Research Journal*, *12*(1), 45–57.

Creswell, J. W. (2011). Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research. In *Pearson* (4th Editio). Pearson.

Demir, Ö., & Seferoglu, S. S. (2021). A Comparison of Solo and Pair Programming in Terms of Flow Experience, Coding Quality, and Coding Achievement. *Journal of Educational Computing Research*, *58*(8), 1448–1466. https://doi.org/10.1177/0735633120949788

Dengler, K., & Matthes, B. (2018). The impacts of digital transformation on the labour market: substitution potentials of occupations in Germany. *Technological Forecasting & Social Change*, *137*, 304–316. https://doi.org/10.1016/j.techfore.2018.09.024

Erol, O., & Çırak, N. S. (2022). The effect of a programming tool scratch on the problem-solving skills of middle school students. *Education and Information Technologies*, *27*(3), 4065–4086. https://doi.org/10.1007/s10639-021-10776-w

Farisiyah, U., Istiyono, E., Hassan, A., Putro, N. H. P. S., Ayriza, Y., Setiawati, F. A., & Mubarok, E. S. (2025). Psychometric properties of the adapted critical language awareness instrument. *Journal of Education and Learning*, *19*(1), 404–415. https://doi.org/10.11591/edulearn.v19i1.21436

Figueiredo, J., & García-Peñalvo, F. J. (2024). Design science research applied to difficulties of teaching and learning initial programming. *Universal Access in the Information Society*, *23*(3), 1151–1161. https://doi.org/10.1007/s10209-022-00941-4

Garcia, M. B. (2021). Cooperative learning in computer programming: A quasi-experimental evaluation of Jigsaw teaching strategy with novice programmers. *Education and Information Technologies*, *26*(4), 4839–4856. https://doi.org/10.1007/s10639-021-10502-6

Gay, L. R., Mills, G. E., & Airasian, P. (2011). *Educational Research: Competencies for Analysis and Applications 10th Edition*. Pearson.

Habib, M. A., Raja-Yusof, R. J., Salim, S. S., Sani, A. A., Sofian, H., & Abu Bakar, A. (2021). Analyzing students' experience in programming with computational thinking through competitive, physical, and tactile games: The quadrilateral method approach. *Turkish Journal of Electrical Engineering and Computer Sciences*, *25*(9), 2280–2297. https://doi.org/10.3906/elk-2010-73

Hai Hom, S. N., & Abdul Talib, C. (2020). Integration of computational thinking skills in teaching and learning programming using the EZ-Prog among matriculation student. *Solid State Technology*, *63*(1s), 670–779.

Hair, J. F., Babin, B. J., & Black, W. C. (2009). *Multivariate data analysis 7th Edition*. Pearson.

Hooshyar, D., Ahmad, R. B., Md Nasir, M. H. N., Shamshirband, S., & Horng, S. J. (2015). Flowchart-based programming environments for improving comprehension and problem-solving skill of novice programmers: A survey. *International Journal of Advanced Intelligence Paradigms*, *7*(1), 24–56. https://doi.org/10.1504/IJAIP.2015.070343

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

Husin, N. F., Mohamad Judi, H., Hanawi, S. A., & Mohd Amin, H. (2020). Technology integration to promote desire to Learn programming in higher education. *International Journal on Advanced Science, Engineering and Information Technology*, *10*(1), 253–259.

Kadar, R., Abdul Wahab, N., Othman, J., Shamsuddin, M., & Mahlan, S. B. (2021). A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review. *International Journal of Academic Research in Progressive Education and Development*, *10*(3), 591–605. https://doi.org/10.6007/ijarped/v10-i3/11100

Kadar, R., Mahlan, S. B., & Shamsuddin, M. (2022). Analysis of Factors Contributing to the Difficulties in Learning Computer Programming among Non- Computer Science Students. *IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 89–94.

Kim, S., & Lee, Y. (2016). The effect of robot programming education on attitudes towards robots. *Technological Forecasting & Social Change*, *9*(24), 1–11. https://doi.org/10.17485/ijst/2016/v9i24/96104

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14–18. https://doi.org/10.1145/1067445.1067453

Lai, A. F., & Yang, C. Y. (2022). The Effectiveness of Block-Based Programming Learning on the Problem-Solving Skills of the Freshmen. *Communications in Computer and Information Science*. https://doi.org/10.1007/978-981-19-9582-8_45

Lai, X., & Wong, G. K. (2022). Collaborative versus individual problem solving in computational thinking through programming : A meta-analysis. *British Journal of Educational Technology*, *53*, 150–170. https://doi.org/10.1111/bjet.13157

Lapawi, N., & Husnin, H. (2020). Investigating students' computational thinking skills on Matter module. *International Journal of Advanced Computer Science and Applications*,

11(11), 310–314. https://doi.org/10.14569/IJACSA.2020.0111140

Laura-Ochoa, L., & Bedregal-Alpaca, N. (2022). Incorporation of Computational Thinking Practices to Enhance Learning in a Programming Course. *International Journal of Advanced Computer Science and Applications*, *13*(2), 194–200. https://doi.org/10.14569/IJACSA.2022.0130224

Lee, Y., & Cho, J. (2020). Knowledge representation for computational thinking using knowledge discovery computing. *Information Technology and Management*, *21*(1), 15–28. https://doi.org/10.1007/s10799-019-00299-9

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Ma, L., Ferguson, J., Roper, M., & Wood, M. (2011). Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education*, *21*(1), 57–80. https://doi.org/10.1080/08993408.2011.554722

Malik, S. I., & Coldwell-Neilson, J. (2017). Impact of a new Teaching and learning approach in an Introductory Programming course. *Journal of Educational Computing Research*, *55*(6), 789–819. https://doi.org/10.1177/0735633116685852

Malik, S. I., Tawafak, R. M., Alfarsi, G., Ashfaque, M. W., & Mathew, R. (2021). A model for enhancing algorithmic thinking in programming education using PAAM. *International Journal of Interactive Mobile Technologies*, *15*(9), 37–51. https://doi.org/10.3991/ijim.v15i09.20617

Mohd Fadzil, A. H., Mohd Nihra Haruzuan, M. S., Noraffandy, Y., & Zaleha, A. (2022). Effects of augmented reality application integration with computational thinking in geometry topics. *Education and Information Technologies*, *27*, 9485–9521. https://doi.org/10.1007/s10639-022-10994-w

Mohd Noor, N. F., Saad, A., & Hashim, A. (2021). Functional requirements of a C-Programming problem-solving application. *Politeknik & Kolej Komuniti Journal of Life Long Learning*, *5*(1), 1–12.

Mohd Rum, S. N., & Zolkepli, M. (2018). Metacognitive strategies in teaching and learning computer programming. *International Journal of Engineering & Technology*, *7*(4.38), 788–794. https://doi.org/10.14419/ijet.v7i4.38.27546

Mohd Yusoff, K., Sahari Ashaari, N., Tengku Wook, T. S. M., & Mohd Ali, N. (2020). Analysis on the requirements of computational thinking skills to overcome the difficulties in learning programming. *(IJACSA) International Journal of Advanced Computer Science and Application*, *11*(3), 244–253.

Mohd Yusoff, K., Sahari Ashaari, N., Tengku Wook, T. S. M., & Mohd Ali, N. (2021). Validation of the Components and Elements of Computational Thinking for Teaching and Learning Programming using the Fuzzy Delphi Method. *International Journal of Advanced Computer Science and Applications*, *12*(1), 80–88. https://doi.org/10.14569/IJACSA.2021.0120111

Morris, H. S., & Liu, S. J. C. (2020). Computational thinking education in the Asian Pacific region. *The Asia-Pacific Education Researcher*, *29*(1), 1–8. https://doi.org/10.1007/s40299-019-00494-w

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. *ACM SIGCSE Bulletin*, *35*(1), 359–362. https://doi.org/10.1145/792548.612006

Namli, N. A., & Aybek, B. (2022). An Investigation of The Effect of Block-Based Programming

and Unplugged Coding Activities on Fifth Graders' Computational Thinking Skills, Self-Efficacy and Academic Performance. *Contemporary Educational Technology*, *14*(1), 1–16. https://doi.org/10.30935/cedtech/11477

Nunnally, J. C., & Bernstein, I. H. (1994). *Psychometric Theory 3rd Edition*. McGraw-Hill. https://doi.org/10.2307/1161962

Ou Yang, F. C., Lai, H. M., & Wang, Y. W. (2023). Effect of augmented reality-based virtual educational robotics on programming students' enjoyment of learning, computational thinking skills, and academic achievement. *Computers and Education*, *195*, 104721. https://doi.org/10.1016/j.compedu.2022.104721

Papadakis, S., Kalogiannakis, M., Zaranis, N., & Orfanakis, V. (2016). Using Scratch and App Inventor for teaching introductory programming in secondary Education. A case study. *International Journal of Technology Enhanced Learning*, *8*, 217–232. https://doi.org/10.1504/ijtel.2016.10001505

Pimpimool, A. (2024). Enhancing Algorithm and Programming Education through Collaborative Blended Learning: A Problem-Based Approach for First-Year Students. *International Journal of Modern Education and Computer Science*, *16*(4), 35–45. https://doi.org/10.5815/ijmecs.2024.04.03

Piwek, P., & Savage, S. (2019). Challenges with learning to program and problem solve: An analysis of student online discussions. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 494–499. https://doi.org/10.1145/3328778.3366838

Polat, E., & Yilmaz, R. M. (2022). Unplugged versus plugged-in: examining basic programming achievement and computational thinking of 6th-grade students. *Education and Information Technologies*, *27*, 9145–9179. https://doi.org/10.1007/s10639-022-10992-y

Rahim, H., Zaman, H. B., Ahmad, A., & Ali, N. M. (2018). Student' s Difficulties in Learning Programming. *Advanced Journal of Technical and Vocational Education*, *2*(3), 40–43. https://doi.org/10.26666/rmp.ajtve.2018.3.7

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Roque-Hernandez, R. V., Guerra-Moya, S. A., & Caballero-Rico, F. C. (2021). Acceptance and assessment in student pair-programming: A case study. *International Journal of Emerging Technologies in Learning*, *16*(9), 4–19. https://doi.org/10.3991/ijet.v16i09.18693

Sabarinath, R., & Quek, C. L. G. (2020). A case study investigating programming students' peer review of codes and their perceptions of the online learning environment. *Education and Information Technologies*, *25*, 3553–3575.

Simon, B., & Hanks, B. (2008). First-year students' impressions of pair programming in CS1. *ACM Journal on Educational Resources in Computing*, *7*(4), 73–85. https://doi.org/10.1145/1316450.1316455

Tan, P. H., Ting, C. Y., & Ling, S. W. (2009). Learning difficulties in programming courses: Undergraduates' perspective and perception. *ICCTD 2009 - 2009 International Conference on Computer Technology and Development*, *1*, 42–46. https://doi.org/10.1109/ICCTD.2009.188

Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers and Education*, *120*, 64–

74. https://doi.org/10.1016/j.compedu.2018.01.011

Toukiloglou, P., & Xinogalos, S. (2024). Effects of Collaborative Support on Learning in Serious Games for Programming. *Journal of Educational Computing Research*. https://doi.org/10.1177/07356331241296888

Ubaidullah, N. H., Mohamed, Z., Hamid, J., & Sulaiman, S. (2021). Improving novice students' computational thinking skills by problem-solving and metacognitive techniques. *International Journal of Learning, Teaching and Educational Research*, *20*(6), 88–108. https://doi.org/https://doi.org/10.26803/ijlter.20.6.5

Watson, C., & Li, F. W. B. (2014). Failure rates in Introductory Programming revisited. *Proceedings Conference on Innovation & Technology in Computer Science Education*, 39–44.

Weng, X., Ng, O. L., Cui, Z., & Leung, S. (2023). Creativity Development With Problem-Based Digital Making and Block-Based Programming for Science, Technology, Engineering, Arts, and Mathematics Learning in Middle School Contexts. *Journal of Educational Computing Research*, *61*(2), 304–328. https://doi.org/10.1177/07356331221115661

Xinogalos, S. (2014). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. *Education and Information Technologies*, *21*(3), 559–588. https://doi.org/10.1007/s10639-014-9341-9

Zhan, Z., Li, T., & Ye, Y. (2024). Effect of jigsaw-integrated task-driven learning on students' motivation, computational thinking, collaborative skills, and programming performance in a high-school programming course. *Computer Applications in Engineering Education*, *32*(6). https://doi.org/10.1002/cae.22793

Zhang, J. H., Meng, B., Zou, L. C., Zhu, Y., & Hwang, G. J. (2023). Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy. *Interactive Learning Environments*, *31*(6), 3792–3809. https://doi.org/10.1080/10494820.2021.1943687

Zheng, L., Zhen, Y., Niu, J., & Zhong, L. (2022). An exploratory study on fade-in versus fade-out scaffolding for novice programmers in online collaborative programming settings. *Journal of Computing in Higher Education*, *19*, 489–516. https://doi.org/10.1007/s12528-021-09307-w

Żyła, K., Chwaleba, K., & Choma, D. (2024). Evaluating Usability and Accessibility of Visual Programming Tools for Novice Programmers—The Case of App Inventor, Scratch, and StarLogo. *Applied Sciences (Switzerland)*, *14*(21). https://doi.org/10.3390/app14219887