Jurnal
Guru Komputer
Journal homepage: https://ejournal.upi.edu/index.php/JGrKom

Student Difficulties in Algorithmic and Programming-Based Computational Thinking: Teachers' Perspectives

Enjun Junaeti[1,], Nusuki Syariati Fatimah[2], Andini Setya Arianti[3]*

Universitas Pendidikan Indonesia, Indonesia
*Correspondence: enjun@upi.edu

ABSTRACT

Integration of computational thinking (CT) into the informatics curriculum at the Junior High School (JHS) level aims to strengthen students' problem-solving abilities. However, its implementation faces various challenges from a pedagogical point of view. This study aims to explore teacher perceptions of the obstacles students face in learning the concepts of algorithms and programming. Using a quantitative approach with a descriptive survey method, data was collected from 37 informatics teachers through a structured questionnaire covering conceptual, procedural, and problem-solving dimensions. The results showed that the conceptual dimension has the highest level of difficulty, especially in materials with high abstraction such as recursion and complex conditional logic. In the procedural dimension, students showed limitations in evaluating and testing algorithms. Meanwhile, in the problem-solving dimension, the main obstacle lies in the decomposition phase and the construction of solutions from open-ended problems. These findings confirm that student difficulties are hierarchical, where a weak understanding of the logical foundation hinders technical and applicative skills.

ARTICLE INFO

Article History:
Submitted/Received 31 Jun 2025
First Revised 10 Jul 2025
Accepted 12 Aug 2025
First Available Online 01 Sep 2025
Publication Date 01 Sep 2025

Keyword:
Algorithm,
Computational Thinking,
Junior High School Teachers,
Learning Difficulties,
Programming.

1. INTRODUCTION

Computer science education has become a fundamental part of the global curriculum in response to massive digital transformation. Computational Thinking (CT) is a problem-solving foundation that includes decomposition, pattern recognition, abstraction, and algorithmic thinking, which is now recognized as an essential skill in informatics learning (Wing, 2006; Tedre & Denning, 2016). At the Junior High School (JHS) level, the integration of computational thinking in algorithmic and programming material aims to build students' conceptual understanding so they can solve problems systematically and logically.

Previous research confirms that computational thinking significantly improves students' problem-solving abilities in various contexts (Grover & Pea, 2013; Rafli et al., 2024). At the secondary education level, the effectiveness of this method is proven to strengthen the understanding of programming structures and algorithm logic (Laura-Ochoa & Bedregal-Alpaca, 2022). In addition, this approach helps students develop structured algorithmic thinking patterns in designing solutions (Anggraini et al., 2024). Although the benefits for students have been widely documented, exploration of implementation challenges from the educators' perspective is still relatively limited (Sentance & Csizmadia, 2017).

There is a literature gap showing that although the benefits of CT have been widely studied, understanding of how teachers identify and mitigate student obstacles in algorithmic concepts is very limited. Teachers play a crucial role in designing adaptive pedagogical strategies for abstract material (Yadav et al., 2016). However, in reality, many educators face difficulties in diagnosing whether students' obstacles lie in conceptual logic aspects or in technical syntactic skills (Lau, 2017). Therefore, research on teacher perspectives is crucial to produce recommendations for more applicable learning strategies.

This study focuses on the analysis of teacher perceptions of student difficulties in integrating computational thinking with algorithm and programming concepts. Several factors identified as major constraints include the complexity of algorithmic logic, students' low initial exposure to computational thinking, and the limitations of learning media capable of visualizing abstract concepts intuitively (Falkner et al., 2019; Qian & Choi, 2023; Faidah, 2023; Yin et al., 2024). By using a survey approach to JHS informatics teachers, this study aims to map the most dominant learning challenge patterns. The results of this study are expected to provide practical contributions to the development of more adaptive informatics curricula and teaching strategies at the secondary school level.

2. METHODS

This study applies a quantitative approach with a descriptive survey method to explore teacher perceptions of student challenges in computational thinking learning integrated with algorithms and programming. A cross-sectional research design was used to collect data from respondents within a certain period of time to provide a systematic overview of the phenomenon studied (Creswell & Creswell, 2018). The research subjects consisted of 37 Informatics teachers at the Junior High School (JHS) level selected through a purposive sampling technique. The inclusion criteria

sample included teachers who actively teach algorithm and programming material, have at least one year of teaching experience, and have implemented computational thinking concepts in the curriculum.

Data was collected using a questionnaire instrument developed based on indicators of programming learning difficulties classified into three main dimensions: understanding of

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

algorithm concepts, procedural skills, and problem-solving abilities based on computational thinking (Fathimah et al., 2025). This instrument consists of 58 closed-ended question indicators based on Fathimah et al. (2025) with a scoring conversion system of 0–3 (Strongly Agree = 3, Agree = 2, Undecided = 1, Disagree = 0). Before distribution, the instrument underwent content validation procedures through expert judgment by three experts in the field of informatics education to ensure the relevance, clarity, and representativeness of each statement item. The reliability of the instrument was tested using Cronbach's Alpha coefficient to ensure the internal consistency of the data (Hair et al., 2019).

The data analysis process was carried out statistically descriptively to identify the main tendencies of the obstacles faced by students. The analysis focused on calculating the mean (mean) and standard deviation values in each dimension and specific indicators. The mean calculation results are then interpreted to determine the relative difficulty level for each learning aspect, from basic concepts to complex algorithmic logic such as recursion and multi-scenario testing. All data is processed systematically to produce adaptive and data-driven teaching strategy recommendations for informatics learning at the secondary school level.

3. RESULTS AND DISCUSSION

3.1. Teacher Demographic Data

Respondents in this study were teachers from various JHS level schools in Indonesia who had taught Informatics subjects in Phase D, especially algorithm and programming material. Based on the questionnaire results, 37 teachers came from various regions such as Bandung, Kupang, Cianjur, Kendari, Tanah Grogot, Majalengka, and Bali. Based on teaching experience and educational background, the distribution of respondents can be seen in **Figures 1a** and **1b**.

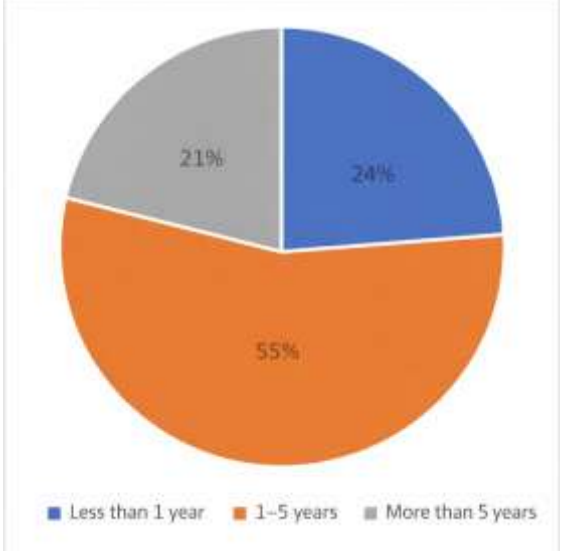

Figure 1a. Teaching experience.

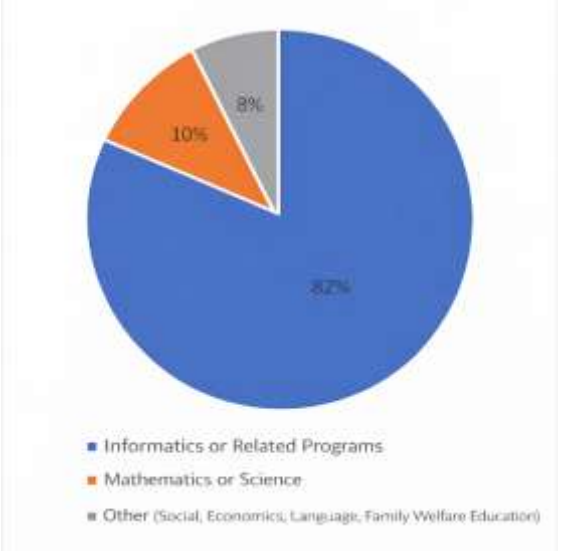

Figure 1b. Educational background.

These findings indicate that most teachers who teach algorithms and programming indeed come from relevant scientific backgrounds, but there are also teachers with non-informatics backgrounds who participate in informatics learning in schools, especially in educational units experiencing limitations in TIK/Informatics teacher resources.

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

The dominance of 1–5 years of teaching experience also reflects that the majority of teachers are in the transition phase from the adaptation phase to the consolidation phase in delivering Computational Thinking (CT) material, especially regarding algorithms and programming. With this background, their perception of student learning difficulties becomes important data in developing targeted learning strategies.

3.2. Student Difficulties in Computational Thinking

The analysis of JHS student algorithm and programming learning difficulties based on teacher perceptions was carried out using the classification of indicators into three main dimensions: understanding of algorithm concepts, programming skills, and problem-solving skills based on computational thinking (Fathimah, 2025). This analysis is based on the average score of teacher perceptions of 58 indicators with a scoring conversion: Strongly Agree = 3, Agree = 2, Undecided = 1, Disagree = 0. The distribution of mean scores for each dimension is visualized in **Figure 2**.

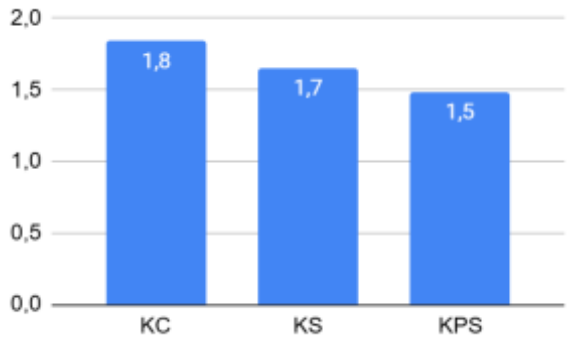

Figure 2. Mean scores of learning difficulties in each learning dimension.

The research findings show that the dimension with the highest level of difficulty is the conceptual understanding aspect, followed by programming skills (procedural), and finally problem-solving skills. The dominance of difficulties in the conceptual aspect confirms that the main obstacle for JHS students in informatics lies in the abstraction of logic which is the basic foundation before entering the technical implementation stage. This phenomenon is in line with Jenkins' (2002) finding which states that programming is a hierarchical skill; the inability to understand basic concepts such as variables and flow control will automatically hinder students' ability to write functional code.

Viewed from a cognitive load perspective, the high difficulty score in the conceptual aspect indicates a challenge in transforming abstract logic into a formal representation. This is in line with the study by Lau (2017) which underlines that students often experience deadlock in the "algorithmic logic" phase before they even deal with syntax errors. However, it is interesting to note that the difficulty score in the problem-solving dimension is relatively lower than the conceptual aspect. This indicates that when students are given clear guidance or problem frameworks, they have the potential to seek solutions, although often constrained by technical instrument limitations or an understanding of the programming language they master.

These findings reinforce that a weak understanding of concepts is the root of further difficulties in writing programs and solving computational problems independently. Therefore, the learning strategies developed must be adaptive and structured. Referring to Vygotsky's scaffolding theory, ideal learning strategies ideally start from strengthening conceptual foundations through gradual approaches, for example by using unplugged

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

computing methods to minimize technical cognitive load at the beginning of learning (Yadav et al., 2016; Dewini et al., 2020). After the conceptual foundation is solid, the process can proceed to the development of technical skills and culminate in application in the context of contextual and authentic problem solving. This integrative approach is expected to reduce the disparity between theoretical understanding and real problem-solving practices in the informatics curriculum at the secondary school level.

3.2.1. Difficulties in the Conceptual Dimension

A deeper exploration of the conceptual dimension shows that the KC22 indicator has the highest difficulty score of 2.24, as illustrated in **Figure 3**. This indicator represents student obstacles in understanding the concept of recursion, which is a function or procedure that calls itself. The high score in this aspect indicates that students experience significant difficulties in processing mental models that demand the ability to think abstractly and hierarchically iterative logic. These findings are in line with the arguments of Yin et al. (2024) who state that recursion is one of the most complex concepts in informatics education because of its non-linear nature and requires a deep understanding of memory execution flow.

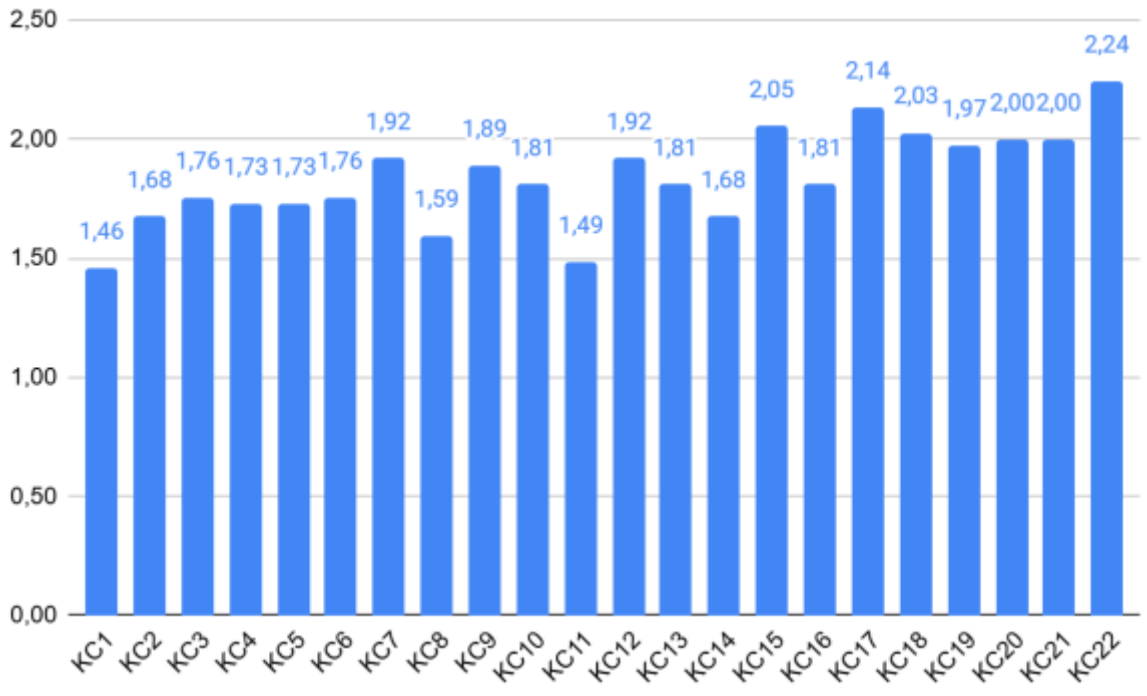

Figure 3. Difficulty scores in the conceptual dimension.

Other indicators with prominent difficulty levels are KC17 (2.14) regarding distinguishing IF and IF–ELSE control structures, as well as a series of indicators KC18 to KC21 covering the use of loops, complex branching, and the implementation of functions and procedures. The difficulty in this aspect indicates a challenge in "conditional logic," where students often fail to map various possibilities for program execution flow. This is in line with the study by Sentance and Csizmadia (2017) which found that mastering control structures is a crucial but difficult transition phase for novice students, because it involves moving from sequential thinking to logical-conditional thinking.

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

Conversely, the indicator with the lowest score was found in KC1 (1.46), which relates to arranging the sequence of instructions from daily problems. This low score proves that concepts that are concrete and contextual are much easier for students to internalize at an early stage. In addition, KC11 (1.49) related to distinguishing data types also showed a relatively low difficulty level. This indicates that JHS students more quickly master aspects of informatics that have direct counterparts in real life or have a simple static structure.

Overall, the pattern in the conceptual dimension shows a positive correlation between the level of material abstraction and students' perception of difficulty. As algorithmic complexity increases—from simple linear sequences to recursion and branching logic—students' cognitive load increases significantly. This phenomenon confirms Cognitive Load Theory in programming learning, where instructions that are too abstract without proper scaffolding can hinder the formation of students' mental schemes (Sin et al., 2025). As a pedagogical implication, these findings demand the application of concrete-to-abstract learning strategies, supported by program visualization media and contextual exercises to bridge students' understanding from real phenomena to formal computational logic.

3.2.2. Difficulties in the Procedural Skills Dimension

The results of the analysis in the procedural skills dimension, as illustrated in Figure 4, reveal that students' main obstacles lie in the ability to evaluate and validate solutions. Indicator KS10 recorded the highest difficulty score of 2.05, which reflects students' ability to evaluate algorithm outputs across various input scenarios. The high score indicates that students experience difficulties in comprehensively testing algorithms. This is reinforced by the score on indicator KS11 (1.92) which shows that even for a single input scenario, the process of evaluating results of algorithms is still a major challenge. This phenomenon is in line with the findings of Falkner et al. (2019), which states that novice students often have fragile mental models of how data flows through algorithms, making it difficult for them to predict results for different input variations.

Furthermore, several indicators, including KS4, KS5, KS7, KS8, KS9, and KS13, exhibit relatively uniform difficulty scores, each approaching 1.8. This series of indicators represents student challenges in composing algorithms that meet both logical (semantic) and notational (syntactic) standards, including technical aspects such as using relevant comments, consistent indentation, and systematic visual writing structure. Difficulty in pouring algorithms into readable pseudocode or flowchart form reflects a low mastery of notation literacy. According to Lau (2017), the inability to follow formal notational standards often hinders the communication of students' algorithmic ideas, which in turn makes the debugging process difficult.

Conversely, indicators KS2 (1.16) and KS6 (1.24) show the lowest difficulty levels in this dimension. KS2 relates to the activity of tracing the simple algorithm process, while KS6 relates to adding basic comments to code lines. The low scores indicate that students are more likely to be able to carry out activities that are observational, mechanical, or documentation-oriented compared to activities that demand the construction of solutions independently. This shows a "construction gap," where students can read or follow an existing flow (tracing), but experience difficulties when they have to build that structure from scratch.

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

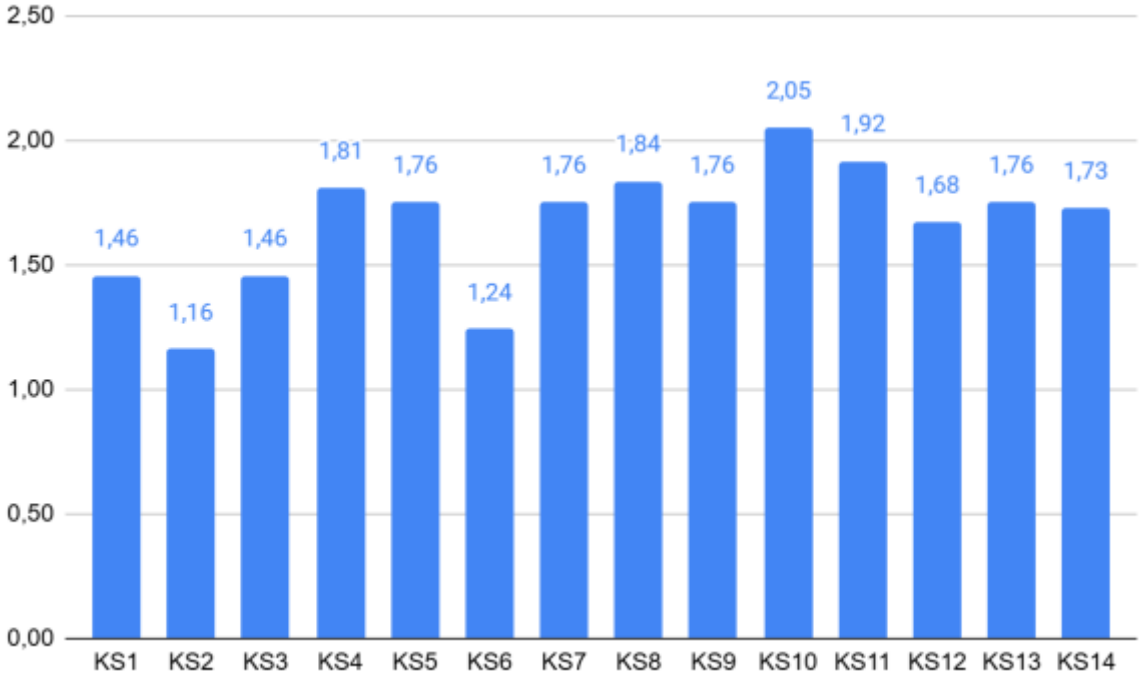

Figure 4. Difficulty scores in the procedural dimension.

Overall, the pattern in the procedural dimension confirms the need for a transformation of learning strategies that emphasize practical aspects and validation. Learning should not just stop at writing syntax, but must include multi-scenario testing exercises as well as strengthening notation standardization and good documentation. The integration of guided debugging exercises is highly recommended to build students' confidence and procedural accuracy in solving computational problems (Grover & Pea, 2013).

3.2.3. Difficulties in the Problem-Solving Dimension

Figure 5 illustrates that the most substantial challenges encountered by students arise in the phases of solution construction and evaluation. Indicators KPS4 through KPS7 display the highest difficulty scores, with values approaching 1.6. These indicators encompass competencies in problem decomposition, the development of algorithmic patterns, and the evaluation of both the correctness and efficiency of the constructed solutions. The high scores on these indicators indicate that students are not yet accustomed to applying the computational thinking framework holistically to construct structured solutions. According to Grover and Pea (2013), the transition from problem understanding to solution construction is a critical stage in CT that requires high-level cognitive skills, so limitations in the decomposition stage often hinder the effectiveness of overall algorithm design.

Conversely, indicators KPS1 and KPS2 show lower difficulty scores (1.2). These two indicators relate to the ability to identify problem input and output, which is part of the abstraction process. The relatively low scores indicate that students are quite familiar with the types of problems that present input data and final results explicitly. However, indicator KPS3 (1.5) shows an increase in difficulty at the transition point, where students are required to transform problem specifications into functional solution designs. This indicates that although students are able to recognize problem components (abstraction), they still experience difficulties in arranging the logical connections between these components (algorithms).

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

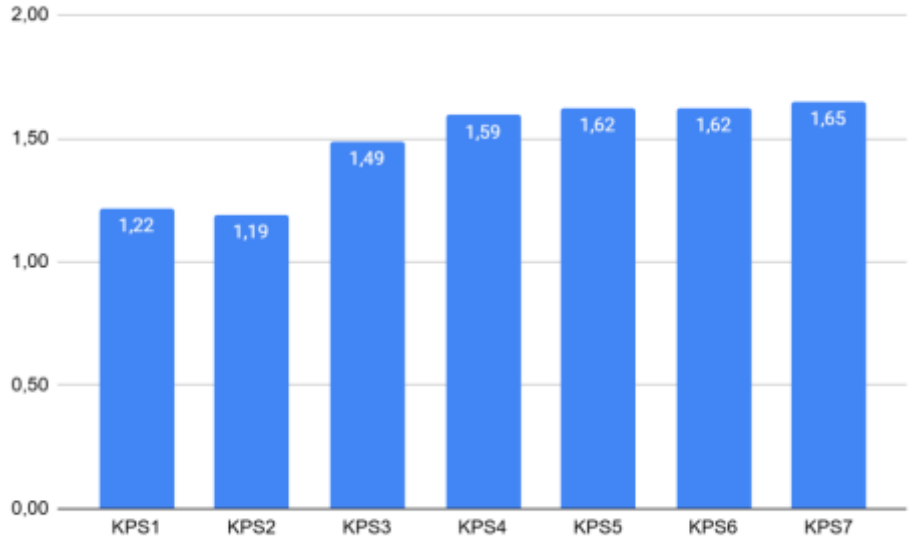

Figure 5. Difficulty scores in the procedural dimension.

In general, the difficulty profile in this dimension reflects students' lack of experience in dealing with complex open-ended problems that require the integration of the pillars of computational thinking. These findings are in line with the arguments of Voon et al. (2022) who emphasize that CT development is not just about understanding concepts, but also about problem-solving practices that demand flexibility of thought. As a managerial implication in learning, the application of Project-Based Learning (PjBL) methods or authentic problem-based learning is needed. This strategy must be able to encourage students to explore various alternative solutions, so they can practice their self-evaluation skills regarding the effectiveness and efficiency of the algorithms they design (Yadav et al., 2016).

4. CONCLUSION

This study concludes that JHS students' main challenges in informatics learning lie in the conceptual dimension, especially in materials with high abstraction such as recursion and complex logical structures. These difficulties are hierarchical, where a weak understanding of the logical foundation hinders students' abilities in the procedural dimension, especially in evaluating output and testing algorithms independently. Meanwhile, in the problem-solving dimension, students are able to perform initial abstraction but still have difficulty deconstructing complex problems into efficient algorithmic solutions.

Overall, this difficulty pattern shows the need for a pedagogical shift from a focus on technical syntax to strengthening mental models through scaffolding strategies. The use of unplugged computing methods to simplify abstract concepts and the implementation of problem-based learning is recommended to increase students' computational thinking flexibility. These findings are expected to be a reference in preparing more adaptive informatics teaching strategies at the secondary school level.

5. ACKNOWLEDGMENT

The author would like to express sincere gratitude to Universitas Pendidikan Indonesia, through the Institute for Research and Community Service (LPPM), for the financial support of this study provided by the Dana Rencana Kerja dan Anggaran Tahunan Penugasan Tahun

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

Anggaran 2025, based on Rector's Decree No. 443/UN40/PT.01.02/2025. This support has been invaluable in enabling the successful completion of this research.

6. AUTHORS' NOTE

The authors declare that there is no conflict of interest regarding the publication of this article. The authors confirmed that the paper was free of plagiarism.

7. REFERENCES

Anggraini, L. M., Nia, K., & Gürbüz, F. (2024). Students' proficiency in computational thinking through constructivist learning theory. *International Journal of Mathematics and Mathematics Education*, *2*(01), 45-59.

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications.

Dewini, D., Rahman, E., & Riza, L. (2021). Penerapan Metode Pembelajaran Computer Science Unplugged (CSU) Pada Mata Pelajaran Administrasi Infrastruktur Jaringan Terhadap Hasil Belajar Siswa (Studi Kasus: SMK N Pekerjaan Umum Bandung). *Jurnal Guru Komputer, 1*(2), 92-99.

Faidah, F. N. (2023). Design and Development of Interactive Multimedia Based on Didactical Design for Basic Programming Subject: Branching Material. *Jurnal Guru Komputer*, *2*(1), 47-56.

Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., ... & Quille, K. (2019, July). An international benchmark study of k-12 computer science education in schools. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 257-258).

Fathimah, N. S. A., Junaeti, E., Aryanti, A. S., & Erlangga, E. (2025). Pengembangan dan Validasi Isi Indikator Kesulitan Belajar Pemrograman Siswa SMP. *Jurnal Evaluasi Pendidikan*, *16*(1), 9-19.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2019). Multivariate data analysis.

Jenkins, T. (2002, August). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (Vol. 4, No. 2002, pp. 53-58).

Lau, W. (2017). *Teaching Computing in Secondary Schools: A Practical Handbook*. Routledge.

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).

Qian, Y., & Choi, I. (2023). Tracing the essence: ways to develop abstraction in computational thinking. *Educational technology research and development*, *71*(3), 1055-1078.

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891

Rafli, M., Sutarno, H., & Wihardi, Y. (2024). Implementation of Computational Thinking in Data Structure Subject Using Problem-based Learning Models. *Jurnal Guru Komputer, 4*(2), 66-75.

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and information technologies*, *22*(2), 469-495.

Shin, Y., Jung, J., Choi, S., & Jung, B. (2025). The influence of scaffolding for computational thinking on cognitive load and problem-solving skills in collaborative programming. *Education and Information Technologies*, *30*(1), 583-606.

Tedre, M., & Denning, P. J. (2016, November). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling international conference on computing education research* (pp. 120-129).

Voon, X. P., Wong, S. L., & Wong, L. H. (2022). Developing computational thinking competencies through constructivist argumentation learning: A problem-solving perspective. *International Journal of Information and Education Technology*.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60*(6), 565–568.

Yin, S. X., Hoe-Lian Goh, D., & Quek, C. L. (2024). Collaborative learning in K-12 computational thinking education: A systematic review. *Journal of Educational Computing Research*, *62*(6), 1220-1254.

DOI: http://dx.doi.org/10.17509/xxxx.xxx
p- ISSN xxxx-xxxx e- ISSN 2774-7891