IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

Programming Self-Efficacy and Perceived Debugging Performance among First-Year Computer Science Students

Daniel S. Saberon[1], Ciberjim L. Aranas[2], Jessica Rose E. Fernandez[3]
Bachelor of Science in Computer Science, College of Computing and Information Sciences
Surigao del Norte State University, Surigao City, Philippines [1,2]
Faculty, College of Computing and Information Sciences
Surigao Del Norte State University, Surigao City, Philippines[3]
daniel.saberon@ssct.edu.ph, cyberjim.aranas@ssct.edu.ph, jfernandez@ssct.edu.ph

Abstract: *This study examined the relationship and level of programming self-efficacy and perceived debugging performance among first-year Computer Science students at Surigao del Norte State University. Using a descriptive-correlational design, 63 student-respondents completed a structured survey measuring four dimensions of programming self-efficacy (coding self-efficacy, problem-solving self-efficacy, programming task persistence, and self-regulated learning in programming) and four dimensions of perceived debugging performance (error identification ability, error correction effectiveness, use of debugging strategies, and debugging autonomy). Results revealed that respondents exhibited high levels across all dimensions of both variables. A consistent positive alignment was observed between programming self-efficacy and perceived debugging performance, indicating that students with higher programming confidence tended to demonstrate better debugging performance in terms of strategy use, error detection and correction, and autonomous debugging. Furthermore, a significant positive relationship was found between programming self-efficacy and perceived debugging performance, as confirmed by Pearson's r analysis.*

Keywords: programming self-efficacy, debugging performance, computer science education, first-year students, descriptive-correlational study

I. INTRODUCTION

Programming is one of the most essential skills in the field of computer science and is one of the first competencies that students are required to learn in their first year. Students of Surigao del Norte State University who study computer science are trained to write, test, and debug computer programs. Debugging, the process of identifying and correcting errors in code is considered one of the most cognitively demanding programming activities for novice computer science students. Programming self-efficacy, the belief a student has in their capability to perform programming tasks successfully, is associated with their debugging strategies and persistence during debugging. As technology-driven industries continue to expand, understanding how and why students develop programming competencies is both important and necessary. Low self-efficacy and poor perceived debugging performance among first-year students may hinder them from developing their programming competencies and from succeeding in programming courses. Therefore, it is important to assess the level of programming self-efficacy and perceived debugging performance of first-year computer science students.

Some recent studies have focused on the importance of self-efficacy in programming learning and its influence on learners' behaviors and outcomes. For instance, Lin et al. (2023) noted that programming self-efficacy is a strong predictor of undergraduate students' effort and persistence in programming, with positive affective experiences acting

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, June 2026 Impact Factor: 8.2

as mediators. Likewise, Zhang et al. (2023) argued that programming self-efficacy is closely associated with the development of computational thinking skills and the ability to manage complex programming problems. With regard to debugging, Yang et al. (2024) conducted a review of the existing literature on the topic, revealing that most novice programmers experience difficulties with debugging because they are unable to apply systematic strategies and because most interventions tend to overlook non-cognitive factors such as self-efficacy. Furthermore, through meta-analysis, Sun et al. (2024) found that structured debugging instruction positively influences debugging performance and accuracy rates. In addition, Shin and Song (2022) stated that providing self-regulated learning support can help improve learners' problem-solving abilities, especially when addressing complex ill-structured programming problems. While all these studies provide insights into self-efficacy in relation to programming and debugging, there are limited studies that examine self-efficacy and its specific dimensions in relation to perceived debugging performance separately. Such an approach is especially needed for first-year computer science students at institutions like Surigao del Norte State University.

Considering the identified gap in the literature, this study was designed to address the specific problem of understanding the relationship between programming self-efficacy and perceived debugging performance among first-year Computer Science students. Students at this level often experienced difficulty in identifying errors, applying effective debugging strategies, and solving programming-related problems independently. These challenges might be associated with differences in students' levels of confidence across various aspects of programming. If such issues were not addressed, students might have developed negative perceptions toward programming, become less motivated, and performed poorly in programming courses.

In order to address the identified problem, this study aimed to examine the relationship between programming self-efficacy and perceived debugging performance based on the analysis of their respective dimensions. In other words, the current study sought to conduct a descriptive study to assess first-year students' levels of coding self-efficacy, problem-solving self-efficacy, programming task persistence, and self-regulated learning in programming, as well as their perceived debugging performance in terms of error identification ability, error correction effectiveness, use of debugging strategies, and debugging autonomy. Through the use of structured Google Forms survey and descriptive statistics, this study assessed and measured the level of programming self-efficacy and perceived debugging performance among first-year computer science students. The results of this research were expected to contribute to educational practices.

Objectives of the Study

This study focused on the connection between programming self-efficacy and debugging performance perception among first-year Computer Science students of Surigao del Norte State University. Specifically, this study aimed to:

1. Determine the demographic profile of first-year Computer Science students at Surigao del Norte State University.
2. Determine the level of programming self-efficacy of first-year Computer Science students in terms of:
   - coding self-efficacy
   - problem-solving self-efficacy
   - programming task persistence
   - self-regulated learning in programming
3. Determine the level of perceived debugging performance of first-year Computer Science students in terms of: perceived error identification ability
   - perceived error correction effectiveness
   - perceived use of debugging strategies
   - perceived debugging autonomy
4. To examine the significant relationship between programming self-efficacy and perceived debugging performance among first-year Computer Science students at Surigao del Norte State University.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, June 2026 Impact Factor: 8.2

5. To assess the levels of each programming self-efficacy dimension across all perceived debugging-performance dimensions among first-year Computer Science students.

II. LITERATURE REVIEW

This chapter presents a synthesis of previous research on programming self-efficacy and perceived debugging performance among first-year Computer Science students. It synthesizes findings from earlier studies by highlighting recurring patterns and relationships, especially those pertaining to the components of programming self-efficacy such as coding self-efficacy, problem-solving self-efficacy, programming task persistence, and self-regulated learning in programming. It also reviews the dimensions of perceived debugging performance, which include error identification, error correction, use of debugging strategies, and debugging autonomy. This review serves as the foundation for establishing the study's variables and analyzing their relationship.

Programming Self-Efficacy in Terms of Coding and Problem-Solving Skills

Coding and problem-solving self-efficacy regularly prove to be among the critical predictors of the abilities of students to perform typical programming activities. While coding self-efficacy refers to learners' belief in their skills related to coding, understanding, and modifying code, problem-solving self-efficacy means that they are capable of solving problems and building algorithms. It can be stated from previous studies that learners who show high levels of confidence in these aspects demonstrate better programming results and make fewer mistakes. For example, according to the findings by Mei et al. (2024), high levels of coding self-efficacy lead to better results in programming courses at the introductory level. Similarly, in their study, Kovari and Katona (2023) proved that self-efficacy in problem-solving was an important indicator of programming skills. In turn, Loksa et al. (2022) argued that learners who were confident used better approaches to coding and had a good analytical mind. Therefore, these two types of self-efficacy can be considered essential aspects influencing students' approaches to performing programming activities, which is necessary for debugging.

Programming Task Persistence and Self-Regulated Learning in Programming

Programming self-efficacy goes beyond the ability to perform technical tasks and includes both behavioral and metacognitive aspects, namely task persistence and self-regulated learning. Task persistence is described by students' ability to persist with a task despite obstacles encountered, whereas self-regulated learning involves planning, monitoring, and assessment of one's learning experience. These aspects have been shown to be crucial for success within programming environments involving trial-and-error procedures and problem solving. For instance, Shin et al. (2023) showed that metacognitive scaffolding leads to greater self-regulated learning and improved performance among students in programming. Likewise, Shin and Song (2022) found higher levels of programming self-efficacy among self-regulated learners and better academic achievements. Finally, Loksa et al. (2022) identified persistence as one of the main factors that helped overcome the challenges of programming.

Perceived Debugging Performance in Terms of Error Identification and Correction

Error detection and error correction are seen as essential components of debugging performance and have been reported to be important skills for novice programmers. Perceived error detection competence can be described as students' perceptions of their skills in detecting errors, while error correction effectiveness is concerned with their skills in correcting identified errors. Literature reports that error detection and correction skills involve not only the application of relevant knowledge but also the use of appropriate strategies. The study conducted by Gao and Hew (2023) revealed that structured debugging training plays an instrumental role in improving learners' error detection and correction abilities. Whalley et al. (2023) similarly report that effective debugging requires analytical and evaluative skills on the part of programmers. Finally, according to the findings presented by Yang et al. (2024), novice programmers find it difficult to detect errors because of insufficient use of appropriate strategies.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, June 2026 Impact Factor: 8.2

Use of Debugging Strategies and Debugging Autonomy

Using appropriate strategies and working independently in solving debugging problems are also key components for effective debugging. Perception of the use of debugging strategies refers to various strategies such as tracing, hypothesis testing, and employing debugging tools. Perception of debugging autonomy refers to the ability to solve problems independently and without any outside help. According to Loksa et al. (2022), the active use of debugging strategies improves the ability of students to solve problems. Gilbert et al. (2024) stated that low self-efficacy results in less independence among students when working on debugging tasks. Moreover, Yusuf and Muhammad (2024) revealed that anxiety prevents learners from implementing appropriate debugging strategies.

Relationship Between Programming Self-Efficacy and Perceived Debugging Performance

An increasing number of studies confirm the connection between programming self-efficacy and debugging effectiveness. The higher the level of students' self-efficacy, the greater the probability of their engagement in efficient troubleshooting activities, persistence in the process of dealing with errors, and use of relevant strategies. According to the results reported by Gilbert et al. (2024), the level of students' debugging self-efficacy affects their persistence and success in problem solving. Shin et al. (2023) also confirmed the influence of self-efficacy on successful problem solving and debugging activities. Moreover, Loksa et al. (2022) noted that students' belief in their capabilities impacts their approaches to debugging. Combining these studies proves that there is a significant link between programming self-efficacy and different dimensions of debugging performance.

Synthesis of Gaps and Research Direction

Despite the extensive amount of empirical evidence suggesting the link between programming self-efficacy and perceived debugging performance, some deficiencies have yet to be addressed. Much research is concerned with assessing overall programming competence without considering specific debugging factors such as error detection, error correction, strategy adoption, and independence (Yang et al., 2024; Whalley et al., 2023). Furthermore, programming self-efficacy is often considered a unitary phenomenon without distinction made between different elements of the concept, including coding self-efficacy, problem-solving self-efficacy, tenacity, and self-regulation (Kovari & Katona, 2023; Shin & Song, 2022). There is also an absence of studies integrating all dimensions of programming self-efficacy and perceived debugging performance into one conceptual model. Most existing investigations are based on laboratory or experimental designs, which may not be appropriate for actual classroom situations, especially among first-year college students (Fu et al., 2022; Loksa et al., 2022). There is therefore a definite need for an all-encompassing study of how certain aspects of programming self-efficacy affect certain aspects of perceived debugging performance.

III. METHODOLOGY

This chapter presents the research methodology adopted in this study to determine the relationship between programming self-efficacy and perceived debugging performance among first-year Computer Science students. It includes the research design, population and sample selection process, data collection instruments, and statistical methods used in the data analysis phase.

Research Design

This study employed a quantitative research design, with a specific focus on a descriptive-correlational approach to determine the level of and significant relationship between programming self-efficacy and perceived debugging performance among first-year Computer Science students at Surigao del Norte State University. A quantitative design was commonly used in educational research since it enabled researchers to quantify variables using a structured instrument and analyze the collected data statistically to ensure the objectivity and validity of the results (Leavy, 2022). The descriptive design was highly appropriate for this study since it allowed the proponents to describe the current

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

status of the variables under investigation (Devi et al., 2022). The descriptive aspect helped establish the demographic profile of the participants and their levels of programming self-efficacy and perceived debugging performance.

Conceptual Framework

This study was guided by Bandura's Self-Efficacy Theory, which suggested that an individual's belief in their own ability to execute behaviors necessary to produce specific performance attainments significantly influenced their actual performance outcomes (Bandura, 1997). Recent scholarly studies continued to validate this framework in computing education contexts, affirming that learners' confidence in their programming capabilities was a robust predictor of persistence, strategy use, and overall academic performance (Kuo et al., 2025; Hsia et al., 2023). Figure 1 showed the key variables, which included Programming Self-Efficacy as the independent variable that influenced Perceived Debugging Performance as the dependent variable.

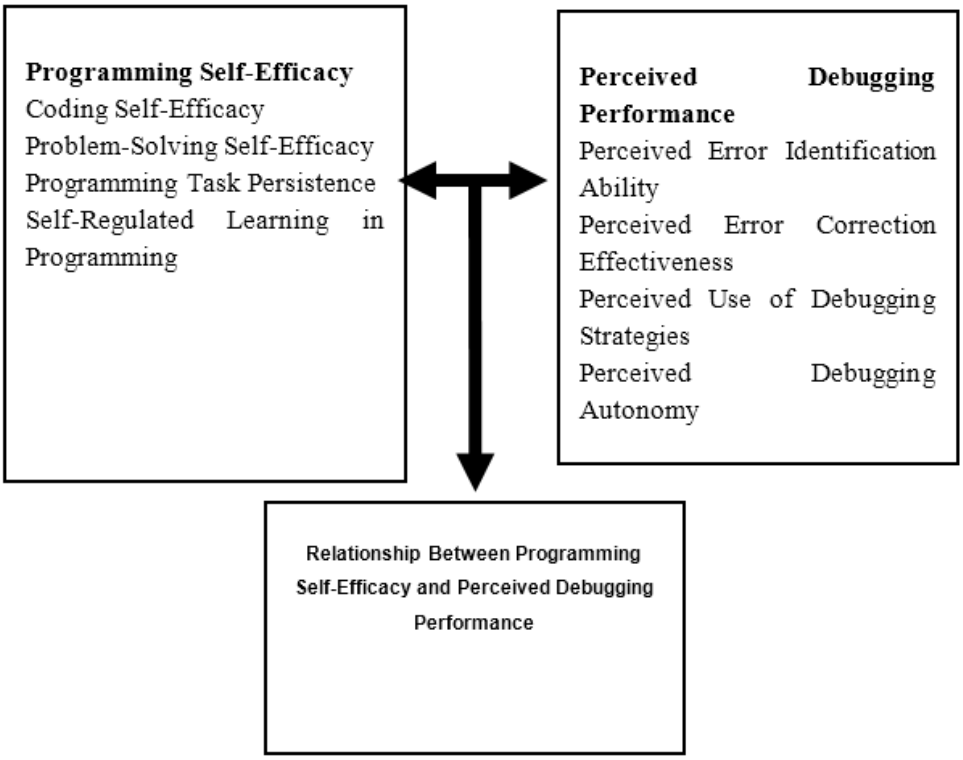

Figure 1. Conceptual Framework of the Study

Programming Self-Efficacy was operationalized through four dimensions: (1) Coding Self-Efficacy, which reflected students' belief in their ability to write syntactically correct code; (2) Problem-Solving Self-Efficacy, which captured their trust in their capacity to logically approach programming problems; (3) Programming Task Persistence, which measured their willingness to continue working through difficult coding challenges; and (4) Self-Regulated Learning in Programming, which assessed their ability to independently acquire new programming knowledge and skills.

These dimensions were hypothesized to have a direct and significant relationship with perceived debugging performance, measured through four indicators: perceived error identification ability, perceived error correction effectiveness, perceived use of debugging strategies, and perceived debugging autonomy. Students with higher levels of programming self-efficacy were expected to demonstrate superior debugging performance, as their confidence, persistence, and autonomy equipped them with the mindset and strategies necessary to effectively identify and resolve code errors (Chen et al., 2025; Yilmaz et al., 2023; Kuo et al., 2025).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

Population and Sampling

The population for this study was composed of all Computer Science students in their first year who had been officially enrolled in the second semester of the school year 2025–2026 at Surigao del Norte State University. The list of enrolled students was retrieved using proper procedures, whereby the proponents requested a letter from the Dean allowing them access to the university registrar's list of enrolled students. From the officially retrieved data, the population of this research comprised 93 first-year Computer Science students.

A non-probability sampling method was utilized, more specifically convenience and voluntary response sampling, as the samples were chosen according to their availability and willingness to participate. Non-probability sampling was commonly practiced in educational studies since samples could not be obtained otherwise due to the need for voluntary participation (Golzar et al., 2022; Singh et al., 2023). The number of participants in the study was 63 students, constituting about 67.74% of the entire population, thus surpassing the minimum response rate of 60% set as the target by the proponents. High response rates were crucial in survey-based studies, as they helped minimize biases due to nonresponse, thereby making results more reliable (Holtom et al., 2022).

The inclusion criteria entailed being registered first-year BSCS students in the particular semester under consideration, whereas the exclusion criteria included individuals who were not on the official registration list or refused to participate in the research.

Data Collection

Data gathering was conducted using a structured survey questionnaire through Google Forms. Before data collection, the proponents sought and obtained official permission from the Dean of Surigao del Norte State University, allowing the study's implementation among first-year BSCS students.

The survey questionnaires were disseminated among all freshman BSCS students through readily available online platforms. The use of online survey tools like Google Forms was common and known to be highly efficient when collecting data in quantitative research (Taherdoost, 2022). Participation was purely voluntary. The survey contained an informed consent section which outlined the research objectives and indicated that participants' data would be kept confidential and anonymous. Data collection took place from April 7, 2026 to April 28, 2026, whereby a total of 63 responses were obtained out of the targeted 93 participants.

Data Analysis

The gathered data were systematically organized, analyzed, and tabulated based on the objectives set for the study. For the first objective, frequency count and percentage distribution were used. For the second and third objectives, a four-point Likert scale was used, and the weighted mean was applied to determine the level of programming self-efficacy and perceived debugging performance. The absence of a neutral point in a four-point scale eliminated central tendency bias and provided more precise results. For the forth objective, Pearson's r was used to determine the significant relationship between programming self-efficacy and perceived debugging performance.

For the fifth objective, descriptive comparison using the weighted mean score and ranking was applied.

The responses were interpreted using the following scale to determine the level of programming self-efficacy and perceived debugging performance:

TABLE I: LIKERT SCALE INTERPRETATION

Scale	Range	Verbal Interpretation
4	3.26 – 4.00	Very High
3	2.51 – 3.25	High
2	1.76 – 2.50	Low
1	1.00 – 1.75	Very Low

DOI: 10.48175/IJARSCT-36510

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

For the forth objective, the Pearson product-moment correlation coefficient (Pearson's r) was employed to determine the significant relationship between programming self-efficacy and perceived debugging performance. The strength and direction of the relationship were interpreted using the following scale:

TABLE IA: Pearson's r Interpretation Scale

r Value	Interpretation
±0.90 – ±1.00	Very High Positive/Negative Correlation
±0.70 – ±0.89	High Positive/Negative Correlation
±0.40 – ±0.69	Moderate Positive/Negative Correlation
±0.20 – ±0.39	Low Positive/Negative Correlation
±0.01 – ±0.19	Very Low Positive/Negative Correlation
0.00	No Correlation

IV. RESULTS AND DISCUSSION

This chapter presents the analysis, interpretation, and discussion of the data gathered from the respondents regarding their programming self-efficacy and perceived debugging performance. The results were organized according to the objectives of the study.

Demographic Profile of First-Year Computer Science Students

As shown in Table II, the highest proportion of the sample consisted of respondents who were 19 years old, making up 73.02% of the total respondents. Following this group were respondents aged 20 years old, constituting 19.05% of the total respondents, while the remainder were respondents aged 21 years old, comprising 7.94% of the total number of respondents. The majority of the participants belonged to the typical age range expected among first-year college students, which implied that the participants experienced comparable transitions as first-year students regarding adaptation to higher education, including introductory computer programming classes (Becker et al., 2023; Whalley et al., 2023; Loksa et al., 2022).

TABLE II: DISTRIBUTION OF RESPONDENTS BY AGE

Age	Frequency	Percentage
19 years old	46	73.02%
20 years old	12	19.05%
21 years old	5	7.94%
Total	**63**	**100.00%**

In Table III, the number of female participants was 35 (55.56%), while males were 28 (44.44%) of the sample. The distribution was quite balanced; however, females constituted the majority of the group, which might suggest emerging shifts in gender dynamics in computing-related courses. This type of sample population was advantageous for conducting research since it helped achieve better representation and enhanced the validity of findings by minimizing potential gender bias (Kotronoulas et al., 2023). Additionally, according to recent literature, efforts to improve gender representation in computing courses had been observed, demonstrating positive trends toward building more inclusive and balanced communities (Perez-Felkner et al., 2025).

TABLE III: DISTRIBUTION OF RESPONDENTS BY GENDER

Gender	Frequency	Percentage
Male	28	44.44%
Female	35	55.56%
Total	63	100.00%

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

Programming Self-Efficacy Across Dimensions

As can be seen from Table IV, all the dimensions of programming self-efficacy were rated as High, which implied that the students had relatively high levels of confidence in their ability to program. Among these dimensions, Coding Self-Efficacy had the highest mean value at 3.056, while Self-Regulated Learning in Programming ranked second at 3.016, followed by Problem-Solving Self-Efficacy at 2.966, and Programming Task Persistence at 2.955.

Having a high level of coding self-efficacy meant that students had strong confidence in their ability to write code, comprehend it, and manipulate it. According to Lishinski (2023), confidence in programming helped students become more engaged in complex programming problems and explore different solutions. Moreover, Loksa et al. (2022) argued that students with high coding self-efficacy tended to persevere in programming in the presence of errors and challenges.

The relatively high mean for self-regulated learning in programming suggested that students had the ability to regulate their learning processes. Self-regulated learners tended to be more effective in programming since they actively sought solutions and adjusted their approach when faced with challenges (Loksa et al., 2022). However, the programming task persistence dimension had the lowest mean value despite its relatively high rating. Research indicated that novice coders often became frustrated due to frequent errors and difficulties in understanding the material (Whalley et al., 2023; Becker et al., 2023). Thus, students demonstrated high self-efficacy in programming but required improvement in terms of persistence.

TABLE IV: LEVEL OF PROGRAMMING SELF-EFFICACY ACROSS DIMENSIONS

Programming Self-Efficacy Dimensions	Weighted Mean	Interpretation
Coding Self-Efficacy	3.056	High
Problem-Solving Self-Efficacy	2.966	High
Programming Task Persistence	2.955	High
Self-Regulated Learning in Programming	3.016	High

Perceived Debugging Performance Across Dimensions

As shown in Table V, all dimensions related to the perception of debugging performance were interpreted as High, meaning that students felt they could effectively perform debugging activities. Of all the dimensions, Perceived Use of Debugging Strategies had the highest mean (3.026), followed by Perceived Error Identification Ability (3.007), Perceived Error Correction Effectiveness (2.987), and Perceived Debugging Autonomy (2.972).

The high level of debugging strategy use suggested that students applied systematic approaches when identifying and resolving errors. Research showed that the use of structured debugging strategies significantly improved programming performance and problem-solving efficiency (Pinto et al., 2023; Whalley et al., 2023; Becker et al., 2023). However, debugging autonomy recorded the lowest mean, which implied that students might still rely on external assistance during debugging. This finding aligned with previous research showing that novice programmers tended to seek guidance from tutors, peers, or the internet whenever they encountered difficulties (Whalley et al., 2023; Prather et al., 2022). Autonomy in debugging typically developed through the gradual accumulation of experience and confidence in performing programming activities (Loksa et al., 2022).

TABLE V: LEVEL OF PERCEIVED DEBUGGING PERFORMANCE ACROSS DIMENSIONS

Perceived Debugging Performance Dimensions	Weighted Mean	Interpretation
Perceived Error Identification Ability	3.007	High
Perceived Error Correction Effectiveness	2.987	High
Perceived Use of Debugging Strategies	3.026	High

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

Perceived Debugging Performance Dimensions	Weighted Mean	Interpretation
Perceived Debugging Autonomy	2.972	High

Significant Relationship Between Programming Self-Efficacy and Perceived Debugging Performance

As presented in Table X, the Pearson's r analysis revealed a significant positive relationship between programming self-efficacy and perceived debugging performance ($r = 0.87$, $p < 0.05$). This indicated a high positive correlation between the two variables, suggesting that as students' programming self-efficacy increased, their perceived debugging performance also tended to increase. These findings were consistent with the theoretical framework of this study, which posited that self-efficacy beliefs directly influenced performance outcomes (Bandura, 1997). Students who held stronger beliefs in their programming capabilities were more likely to engage in effective debugging behaviors, persist through errors, and employ systematic debugging strategies. The significant positive relationship found in this study supported and extended the findings of Gilbert et al. (2024), who reported that students with higher debugging self-efficacy demonstrated greater persistence and success in problem solving, and Shin et al. (2023), who confirmed the influence of self-efficacy on successful debugging activities. Furthermore, the result corroborated the observation of Loksa et al. (2022) that students' belief in their capabilities shaped their approaches to debugging. These converging pieces of evidence affirmed that programming self-efficacy was not merely associated with general programming proficiency but also served as a meaningful predictor of perceived debugging performance among first-year Computer Science students.

TABLE VI: PEARSON'S R: RELATIONSHIP BETWEEN PROGRAMMING SELF-EFFICACY AND PERCEIVED DEBUGGING PERFORMANCE

Variables	Pearson's r	p-value	Interpretation
Programming Self-Efficacy → Perceived Debugging Performance	0.87	<0.05*	High Positive Correlation – Significant

*$p < 0.05$ (significant)

Relationship Between Programming Self-Efficacy and Perceived Debugging Performance Dimensions

As presented in Tables VII to X, all the dimensions of perceived debugging performance with regard to all the programming self-efficacy variables were assessed as High. There was a consistent trend between these two measures. In Table VII, when coding self-efficacy was analyzed, it could be seen that the highest mean value pertained to Perceived Use of Debugging Strategies (3.111). Students who felt competent in coding tended to use more effective strategies for debugging, as they understood the structure of the program and were therefore able to easily identify and fix bugs (Lishinski, 2023; Becker et al., 2023).

TABLE VII: LEVEL OF PERCEIVED DEBUGGING PERFORMANCE BY CODING SELF-EFFICACY

Coding Self-Efficacy	Weighted Mean	Interpretation
Perceived Error Identification Ability	3.069	High
Perceived Error Correction Effectiveness	3.048	High
Perceived Use of Debugging Strategies	3.111	High
Perceived Debugging Autonomy	2.995	High

In Table VIII, for problem-solving self-efficacy, the highest mean was observed in Perceived Debugging Autonomy (3.026). Students who had self-confidence regarding their problem-solving capabilities tended to debug in a more autonomous manner. Problem-solving self-confidence supported decision-making and analysis (Lishinski & Yadav,

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

2021) and minimized dependence on external sources, as students believed they could solve problems independently (Whalley et al., 2023).

TABLE VIII: LEVEL OF PERCEIVED DEBUGGING PERFORMANCE BY PROBLEM-SOLVING SELF-EFFICACY

Problem-Solving Self-Efficacy	Weighted Mean	Interpretation
Perceived Error Identification Ability	2.995	High
Perceived Error Correction Effectiveness	2.905	High
Perceived Use of Debugging Strategies	2.937	High
Perceived Debugging Autonomy	3.026	High

In Table IX, for programming task persistence, all factors maintained high levels, with debugging autonomy having the lowest mean. Persistence motivated students to continue working on their tasks but could not ensure complete autonomy. Students who persisted continued working despite challenges (Prather et al., 2022), but persistence needed to be combined with appropriate techniques to achieve desirable results (Loksa et al., 2022).

TABLE IX: LEVEL OF PERCEIVED DEBUGGING PERFORMANCE BY PROGRAMMING TASK PERSISTENCE

Programming Task Persistence	Weighted Mean	Interpretation
Perceived Error Identification Ability	3.000	High
Perceived Error Correction Effectiveness	2.989	High
Perceived Use of Debugging Strategies	2.958	High
Perceived Debugging Autonomy	2.873	High

In Table X, under self-regulated learning, the highest mean was observed in Perceived Use of Debugging Strategies (3.101). This indicated that students who effectively managed their learning process had a more strategic approach to debugging. Learners who regulated themselves tended to plan, monitor, and evaluate their actions, thus becoming more systematic in solving problems (Loksa et al., 2022; Prather et al., 2022; Lishinski & Yadav, 2021).

TABLE X: LEVEL OF PERCEIVED DEBUGGING PERFORMANCE BY SELF-REGULATED LEARNING IN PROGRAMMING

Self-Regulated Learning in Programming	Weighted Mean	Interpretation
Perceived Error Identification Ability	2.963	High
Perceived Error Correction Effectiveness	3.005	High
Perceived Use of Debugging Strategies	3.101	High
Perceived Debugging Autonomy	2.995	High

Overall, the consistent High ratings across Tables VI to IX indicated a strong alignment between programming self-efficacy and perceived debugging performance. These findings supported the theoretical framework according to which self-efficacy influenced performance outcomes. Learners who had confidence in themselves tended to participate actively, persevere despite difficulties, and employ appropriate methods (Loksa et al., 2022; Lishinski & Yadav, 2021; Lishinski, 2023).

V. CONCLUSION

In conclusion, this study assessed the level and relationship between programming self-efficacy and perceived debugging performance of first-year Computer Science students at Surigao del Norte State University. It was found that

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, June 2026 Impact Factor: 8.2

the respondents demonstrated high levels of programming self-efficacy across all four dimensions, including coding self-efficacy, problem-solving self-efficacy, programming task persistence, and self-regulated learning in programming. In addition, the students exhibited high levels of perceived debugging performance along the dimensions of error identification, effectiveness of error correction, implementation of debugging strategy, and debugging autonomy. Furthermore, through all the dimensions of programming self-efficacy and perceived debugging performance, high levels of programming self-efficacy were correlated with high levels of perceived debugging performance, thereby revealing a positive relationship between the two variables. These results implied that students who were more competent in programming had higher perceived debugging performance in terms of strategy implementation, error detection and correction, and debugging autonomy.

VI. RECOMMENDATION

Considering the findings of this study, it is recommended that future research evaluate the performance of participants in debugging by using hands-on programming tests rather than relying solely on their own perceptions. In addition, further research should be conducted to investigate other factors affecting debugging performance, such as programming experience, teaching methods, and learning environment. For educators, it will be useful to implement interventions focused on enhancing programming self-efficacy, particularly in problem-solving and persistence, which may positively affect debugging ability. Conducting more research with larger groups across different educational institutions would also lead to better generalization of the obtained results. It would also be beneficial to investigate whether specific teaching techniques help in improving programming self-efficacy and debugging ability among Computer Science students.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the Dean and faculty of the College of Computing and Information Sciences at Surigao del Norte State University for granting permission to conduct this study. Special thanks are extended to all first-year Computer Science students who willingly participated in the survey. The authors also acknowledge the guidance and support received throughout the research process.

REFERENCES

1. Bandura, A. (1997). Self-efficacy: The exercise of control. W. H. Freeman and Company.
2. Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). 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, 500–506. https://doi.org/10.1145/3545945.3569759
3. Chen, S. J., Shan, X., Liu, Z. M., & Chen, C. Q. (2025). Employing large language models to enhance K-12 students' programming debugging skills, computational thinking, and self-efficacy. Educational Technology & Society, 28(2), 259–278. https://www.jstor.org/stable/48827986
4. Devi, B., Devi, R., Pradhan, S., Giri, D., Lepcha, N., & Basnet, S. (2022). Application of correlational research design in nursing and medical research. Journal of Xi'an Shiyou University, Natural Sciences Edition, 65(11). https://doi.org/10.17605/OSF.IO/YRZ68
5. ramming environments. Interactive Learning Environments. https://doi.org/10.1080/10494820.2022.2146141
6. Gao, X., & Hew, K. F. (2023). A flipped systematic debugging approach to enhance elementary students' program debugging performance and optimize cognitive load. Journal of Educational Computing Research, 61(5), 1064–1095. https://doi.org/10.1177/07356331221133560
7. Gilbert, C., McDonald, B., & Scott, M. J. (2024). Exploring the relationship between debugging self-efficacy and CASE tools for novice troubleshooting. UKICER '24: Proceedings of the 2024 Conference on United Kingdom & Ireland Computing Education Research. ACM. https://doi.org/10.1145/3689535.3689556

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, June 2026 Impact Factor: 8.2

8. Golzar, J., Noor, S., & Tajik, O. (2022). Convenience sampling. International Journal of Education & Language Studies, 1(2), 72–77. https://doi.org/10.22034/ijels.2022.162981
9. Holtom, B., Baruch, Y., Aguinis, H., & Ballinger, G. A. (2022). Survey response rates: Trends and a validity assessment framework. Human Relations, 75(8), 1560–1584. https://doi.org/10.1177/00187267211070769
10. Hsia, L. H., Lin, Y. N., & Hwang, G. J. (2023). What drives undergraduates' effort and persistence in learning programming. Education and Information Technologies, 28, 11507–11530. https://doi.org/10.1007/s10639-023-
11. Kotronoulas, G., Miguel, S., Dowling, M., Fernández-Ortega, P., Colomer-Lahiguera, S., Bağçivan, G., Pape, E., Drury, A., Semple, C., Dieperink, K. B., & Papadopoulou, C. (2023). An overview of the fundamentals of data management, analysis, and interpretation in quantitative research. Seminars in Oncology Nursing, 39(2), Article 151398. https://doi.org/10.1016/j.soncn.2023.151398
12. Kovari, A., & Katona, J. (2023). Effect of software development course on programming self-efficacy. Education and Information Technologies, 28, 10937–10963. https://doi.org/10.1007/s10639-023-11617-8
13. Kuo, Y. T., & Kuo, Y. C. (2025). Learning programming: Exploring the relationships of self-efficacy, computational thinking, and learning performance among minority students. Frontiers in Education, 10, 1623415. https://doi.org/10.3389/feduc.2025.1623415
14. Kusmaryono, I., Wijayanti, D., & Maharani, H. R. (2022). Number of response options, reliability, validity, and potential bias in the use of the Likert scale education and social science research: A literature review. International Journal of Educational Methodology, 8(4), 625–637. https://doi.org/10.12973/ijem.8.4.625
15. Leavy, P. (2022). Research design: Quantitative, qualitative, mixed methods, arts-based, and community-based
16. Lin, G. Y., Liao, Y. W., Su, Z. Y., Lai, C. F., & Tsai, C. C. (2023). What drives undergraduates' effort and persistence in learning programming. Education and Information Technologies, 28, 12383–12406. https://doi.org/10.1007/s10639-023-11670-3
17. Lishinski, A. (2023). Self-efficacy feedback loops and learning experiences in CS1. Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, 1–7. https://doi.org/10.1145/3587102.3588837
18. Lishinski, A., & Yadav, A. (2021). Self-evaluation interventions: Impact on self-efficacy and performance in introductory programming. ACM Transactions on Computing Education, 21(3), 1–28. https://doi.org/10.1145/3446199
19. Loksa, D., Margulieux, L., Becker, B. A., Craig, M., Denny, P., Pettit, R., & Prather, J. (2022). Metacognition and self-regulation in programming education: Theories and exemplars of use. ACM Transactions on Computing Education, 22(4), Article 39. https://doi.org/10.1145/3487050
20. Mei, Y., Ping, H., & Shen, M. (2024). The effect of self-efficacy and pair programming experience in learning results of introductory programming courses. arXiv preprint. https://doi.org/10.48550/arXiv.2410.15558
21. Mishra, P., Pandey, C. M., Singh, U., Gupta, A., Sahu, C., & Keshri, A. (2019). Descriptive statistics and normality tests for statistical data. Annals of Cardiac Anaesthesia, 22(1), 67–72. https://doi.org/10.4103/aca.ACA_157_18
22. Perez-Felkner, L., Erichsen, K., Li, Y., Chen, J., Hu, S., Ramirez Surmeier, L., & Shore, C. (2025). Computing education interventions to increase gender equity from 2000 to 2020: A systematic literature review. Review of Educational Research, 95(3), 536–580. https://doi.org/10.3102/00346543241241536
23. Pinto, J. D., Zhang, Y., Paquette, L., & Fan, A. X. (2023). Investigating the relationship between programming experience and debugging behaviors in an introductory computer science course. International Conference on Quantitative Ethnography (ICQE 2023), Communications in Computer and Information Science, 1895, 125–139. https://doi.org/10.1007/978-3-031-47014-1_9

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, June 2026
Impact Factor: 8.2

24. Prather, J., Margulieux, L., Whalley, J., Denny, P., Reeves, B. N., Becker, B. A., Singh, P., Powell, G., & Bosch, N. (2022). Getting by with help from my friends: Group study in introductory programming understood as socially shared regulation. Proceedings of the 2022 ACM Conference on International Computing Education Research V. 1, 1–14. https://doi.org/10.1145/3501385.3543970
25. Robie, C., Meade, A. W., Risavy, S. D., & Rasheed, S. (2022). Effects of response option order on Likert-type psychometric properties and reactions. Educational and Psychological Measurement, 82(6), 1107–1129. https://doi.org/10.1177/00131644211069406
26. Shin, Y., & Song, D. (2022). The effects of self-regulated learning support on learners' task performance and cognitive load in computer programming. Journal of Educational Computing Research, 60(6), 1490–1513. https://doi.org/10.1177/07356331211052632
27. Shin, Y., Jung, J., Zumbach, J., & Yi, E. (2023). The effects of worked-out example and metacognitive scaffolding on problem-solving programming. Journal of Educational Computing Research, 61(6), 1312–1331. https://doi.org/10.1177/07356331231174454
28. Singh, P., Kaur, H., Kaur, A., & Bhardwaj, P. (2023). Clinical research: A review of study designs, hypotheses, errors, sampling types, ethics, and informed consent. Cureus, 15(1), e33374. https://doi.org/10.7759/cureus.33374
29. Sun, C., Yang, S., & Becker, B. (2024). Debugging in computational thinking: A meta-analysis on the effects of interventions on debugging skills. Journal of Educational Computing Research, 62(4), 867–901. https://doi.org/10.1177/07356331241227793
30. Taherdoost, H. (2022). Designing a questionnaire for a research paper: A comprehensive guide to design and develop an effective questionnaire. Asian Journal of Managerial Science, 11(1), 8–16. https://doi.org/10.51983/ajms-2022.11.1.3087
31. Tyumeneva, Y., Sudorgina, Y., Kislyonkova, A., & Lebedeva, M. (2022). Ordering motivation and Likert scale ratings: When a numeric scale is not necessarily better. Frontiers in Psychology, 13, 942593. https://doi.org/10.3389/fpsyg.2022.942593
32. Whalley, J., Settle, A., & Luxton-Reilly, A. (2023). A think-aloud study of novice debugging. ACM Transactions on Computing Education, 23(2), 1–38. https://doi.org/10.1145/3589004
33. Yang, S., Baird, M., O'Rourke, E., Brennan, K., & Schneider, B. (2024). Decoding debugging instruction: A systematic literature review of debugging interventions. ACM Transactions on Computing Education, 24(4), 1–44. https://doi.org/10.1145/3690652
34. Yilmaz, R., & Yilmaz, F. G. K. (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, 100147. https://doi.org/10.1016/j.caeai.2023.100147
35. Yusuf, A., & Muhammad, A. Y. (2024). Exploring clusters of novice programmers' anxiety-induced behaviors during coding and debugging. Journal of Educational Computing Research. https://doi.org/10.1177/07356331241270707
36. 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