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 Relationship Between Programming Practice Hours and Problem-Solving Skills of Computer Science Students

Julie Ann C. Saberon[1], Jozavel B. Matiga[2], Jessica Rose E. Fernandez[3]
BSCS Students, 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]
saberonjulieann82@gmail.com matigajozavel@gmail.com jfernandez@ssct.edu.ph

Abstract: *Programming has become an essential skill in the field of technology and computer science. This study explored the relationship between the number of hours spent on programming practice and the problem-solving skills of Computer Science students at Surigao del Norte State University. A quantitative descriptive-correlational research design was used to determine whether the amount of time students dedicated to coding activities was related to their ability to solve problems effectively. Data were collected from 54 Computer Science students through a structured questionnaire distributed using Google Forms. The responses were then analyzed using descriptive statistical methods. The results showed that students who spent more time practicing programming generally demonstrated better problem-solving abilities. These students performed well in areas such as analytical thinking, logical reasoning, and developing solutions to programming-related tasks. The findings indicate that regular involvement in coding activities can help improve the way students analyze and address challenges. Based on the results, the study concluded that consistent programming practice plays an important role in enhancing problem-solving skills and increasing students' confidence in handling complex programming tasks. Therefore, it is recommended that students participate in regular coding exercises and hands-on programming activities to further strengthen their technical knowledge and critical thinking skills.*

Keywords: Programming Practice, Problem-Solving Skills, Computer Science Students, Programming Education, Quantitative Research

I. INTRODUCTION

In today's digital age, programming has become one of the most important skills for students pursuing careers in technology and computer science. Aside from being essential in software development, programming also helps students develop logical thinking, analytical skills, and the ability to solve problems effectively. As technology continues to advance, students need to build not only their technical knowledge but also their critical thinking skills to keep up with the demands of the industry.

This study focused on determining the relationship between programming practice hours and problem-solving skills among Computer Science students at Surigao del Norte State University. Although students now have access to various programming resources, tutorials, and learning platforms, many still experience difficulties when dealing with challenging programming tasks. Some students find it hard to identify errors in their code, develop effective solutions, or apply programming concepts in actual situations. Because of these challenges, the researchers became interested in finding out whether the amount of time spent practicing programming has an influence on students' problem-solving

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

abilities.

Previous studies have shown that regular practice plays an important role in learning programming. According to Malik and Coldwell-Neilson (2022), consistent and structured coding activities help students improve their computational thinking and problem-solving skills. Ansong-Gyimah (2022) also pointed out that students who frequently engage in hands-on programming activities tend to have stronger analytical skills and a better understanding of programming concepts than those who rely mainly on lectures or reading materials. In addition, Rodríguez-Martínez et al. (2022) identified lack of practice as one of the common reasons why students struggle in programming courses. Denny et al. (2023) further emphasized that active coding exercises and problem-based learning activities can help students understand programming concepts more effectively and perform better in programming-related tasks.

Despite the availability of studies related to programming education, only a few have specifically examined the connection between programming practice hours and problem-solving skills among Computer Science students in local institutions such as Surigao del Norte State University. Most previous research focused on teaching methods, curriculum development, or students' overall programming performance. Because of this gap, the researchers believed that conducting a study on this topic would provide useful insights and contribute to existing knowledge.

The main goal of this study was to determine whether spending more time on programming practice could help improve students' problem-solving skills. The findings may benefit students, teachers, and academic institutions by highlighting the importance of consistent practice in programming education. Furthermore, the results may serve as a basis for developing strategies and activities that can strengthen students' programming abilities, critical thinking skills, and preparedness for future academic and professional challenges.

OBJECTIVES OF THE STUDY

The study aimed to:

1. Examine the impact of programming practice hours on the problem-solving skills of Computer Science students.
2. Determine the relationship between programming practice hours and problem-solving skills.
3. Identify the effectiveness of regular programming practice in improving students' problem-solving abilities.
4. Determine the common challenges encountered by students in developing problem-solving skills in programming.
5. Propose recommendations to improve programming practice and problem-solving skills among students.

Conceptual Framework

This study was anchored on Ericsson's Theory of Deliberate Practice (2022), which explains that skills are developed through consistent, focused, and repeated practice over time. In the context of programming, this means that students can improve their problem-solving abilities through continuous engagement in coding tasks and algorithm exercises. In this study, Programming Practice Hours served as the independent variable, representing how much students actively participate in programming activities. It was measured using indicators such as weekly practice hours, coding consistency, frequency of

practice, and involvement in algorithm exercises. These indicators helped describe how students allocate their time, maintain regular coding habits, and actively engage in programming tasks.

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

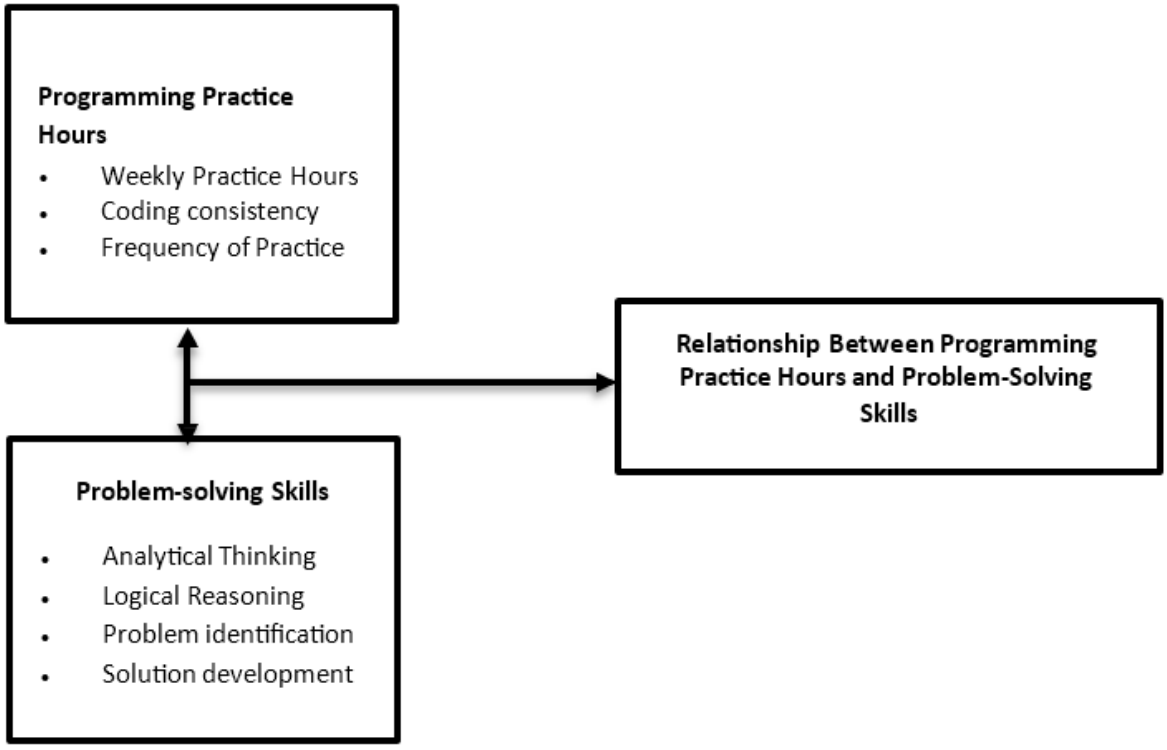

Figure 1. Conceptual Framework of the Study

II. REVIEW OF RELATED LITERATURE

This section presents related studies and literature that are relevant to the current research. The reviewed works mainly discuss programming practice, problem-solving skills, cognitive development, student engagement, learning strategies, and academic performance in computer science education. Together, these studies help explain how regular practice in coding may affect students' learning, improve their technical abilities, and contribute to their overall performance in programming subjects.

Programming Practice and Problem-Solving Skills

Shin et al. (2023) looked into how worked-out examples and metacognitive scaffolding affect the programming problem-solving skills of beginner programmers. Their findings showed that students who were given guided practice and well-structured programming activities tended to perform better when solving problems. They also developed stronger self-regulation skills along the way. The study highlighted that when students consistently practice programming with proper guidance, they gain a clearer understanding of key concepts and experience fewer learning difficulties.

Quadir et al. (2023) focused on the different factors that influence programming problem-solving skills among computer science students. The results of their study revealed that frequent programming practice and motivational learning strategies had a positive impact on students' performance in coding tasks. It was also observed that students who regularly engaged in hands-on coding activities became more confident in tackling programming problems and generally achieved better academic results.

Eckerdal, Berglund, and Thune (2023) examined the role of programming theory alongside practical laboratory activities in shaping students' learning experiences. They found that when theoretical lessons were combined with

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

actual coding practice, students gained a deeper understanding of programming concepts and improved their technical abilities. Their study emphasized that laboratory-based activities gave students the opportunity to apply what they learned in real coding situations, helping to strengthen their logical reasoning and analytical thinking skills.

Malik and Coldwell-Neilson (2022) explored how deliberate practice contributes to the development of computational thinking among undergraduate programming students. Their results showed that learners who consistently engaged in structured and goal-oriented coding exercises over time performed better in problem-solving compared to those who practiced only occasionally. The study emphasized that both the consistency and quality of practice play an important role in improving programming skills.

Denny et al. (2023) investigated the use of problem-based learning in introductory programming courses. They found that students who were regularly given real-world coding problems were able to improve their debugging skills, strengthen their understanding of algorithms, and gain more confidence when dealing with unfamiliar programming tasks. Overall, the study suggested that problem-based learning helps produce more capable and flexible student programmers.

Rodríguez-Martínez et al. (2022) identified lack of sufficient practice as a major reason for student difficulties in programming subjects. Their multi-university study revealed that students who spent more time practicing coding outside the classroom performed better in both written exams and hands-on assessments. Because of this, the researchers recommended adding structured programming activities beyond regular class hours to support learning.

Luo et al. (2023) examined how self-directed programming practice affects the academic performance of first-year computer science students. The findings showed that students who willingly practiced coding beyond the required coursework achieved higher scores in problem-solving tests and felt more confident in their programming abilities. The study highlighted the importance of encouraging students to take initiative in practicing programming on their own.

Ansong-Gyimah (2022) studied the effects of hands-on programming activities on students' analytical and logical thinking skills. The results indicated that students who regularly engaged in practical coding tasks developed a clearer and more systematic way of breaking down and solving problems. The study recommended that programming courses should include more practical exercises to help strengthen higher-order thinking skills.

Cetin and Ozden (2022) focused on how regular coding practice affects beginners' ability to trace and debug programs. Their findings showed that students who spent more time practicing were much better at identifying and fixing errors in their code compared to those with limited practice. The researchers concluded that consistent debugging practice is an effective way to improve overall programming problem-solving skills.

Student Engagement and Academic Performance

Trajkovic et al. (2026) looked into the connection between student engagement and academic performance in science and technical subjects. Their findings showed that students who regularly practiced programming on their own tended to get higher scores in technical assessments compared to those who were less engaged. The study pointed out that consistent involvement and active participation in programming tasks play a big role in improving both performance and skill development.

Orr (2023) studied the use of pair programming and problem-solving studios in computer science education. The results indicated that students who worked in pairs or collaborative settings became more engaged and motivated in learning programming. They also showed a better understanding of coding tasks and gained more confidence in solving problems. The study highlighted that teamwork and immediate feedback can significantly improve programming learning outcomes.

Hayashi and Nakano (2022) examined student activity on online programming platforms and its relationship to academic performance. They found that students who frequently logged in, completed exercises, and reviewed feedback consistently achieved better grades and demonstrated stronger problem-solving abilities. The researchers noted that online learning tools can effectively support traditional classroom instruction when used regularly.

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

Wang et al. (2023) investigated what motivates computer science students to engage in voluntary programming practice. Their study revealed that internal motivation, the satisfaction of solving coding problems, and encouragement from peers were the main reasons students continued practicing. The researchers emphasized that a positive and supportive learning environment is important in encouraging students to practice programming more often.

Loksa et al. (2022) focused on how self-regulation strategies affect programming performance among university students. The results showed that students who set their own goals, tracked their progress, and reflected on their mistakes were more engaged and performed better in problem-solving tasks. The study highlighted that developing self-awareness in learning is just as important as improving technical programming skills.

Santos et al. (2023) explored classroom participation in a flipped learning setup and its effect on programming performance. They found that students who actively participated in in-class coding activities and completed preparatory tasks before class generally performed better in programming exams than those who were less involved. The study concluded that active engagement both inside and outside the classroom strongly contributes to programming success.

Hermans and Aivaloglou (2022) examined how early exposure to programming affects long-term learning and engagement. Their findings suggested that students who were introduced to coding at an earlier stage felt more comfortable with programming tasks and performed better in advanced courses later on. The study recommended introducing programming concepts earlier in education to help students build stronger foundational skills.

Cognitive Development and Computational Thinking

Kafai and Burke (2022) reviewed the cognitive advantages of learning programming across different educational levels. They found that programming helps develop computational thinking skills such as breaking down problems, recognizing patterns, creating abstractions, and designing algorithms. The study concluded that these skills are not limited to coding alone but can also be applied to solving real-life problems in different contexts.

Tang et al. (2023) explored how programming education contributes to higher-order thinking skills among university students. Their findings showed that students who regularly engaged in coding activities improved in areas such as critical thinking, creativity, and logical reasoning. The researchers pointed out that programming is not only a technical skill but also a way to strengthen overall intellectual ability.

Kong et al. (2022) examined the link between computational thinking and problem-solving performance among both secondary and tertiary students. The results revealed that students with stronger computational thinking skills were better at analyzing complex problems, identifying patterns, and developing organized solutions. The study suggested that computational thinking should be integrated more deeply into programming lessons to better prepare students academically and professionally.

Weintrop et al. (2022) looked at how block-based and text-based programming environments affect students' problemsolving skills. They found that students who transitioned from visual programming tools to text-based coding showed improvements in both technical ability and abstract thinking. The study emphasized that a gradual progression in programming tools can help students develop deeper understanding and stronger cognitive skills.

Atmatzidou and Demetriadis (2022) investigated the impact of robotics-based programming activities on undergraduate students' computational thinking. Their findings showed that students involved in robotics projects improved in logical sequencing, pattern recognition, and algorithmic thinking. The study suggested that hands-on, project-based programming experiences create a more meaningful environment for developing cognitive skills.

Programming Assessment and Learning Support

A 2024 study on code comprehension and programming assessment among beginner programmers found that clear feedback and well-designed assessment methods helped students better understand programming concepts and improve their coding performance. It also highlighted the importance of effective evaluation tools in tracking student progress and identifying areas that need further support.

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

Scholl and Kiesler (2024) studied the increasing use of artificial intelligence tools like ChatGPT in programming education. They found that many beginner programmers rely on these tools for help in understanding concepts and solving coding tasks. While AI tools can support learning, the study emphasized that students still need strong critical thinking and independent problem-solving skills to fully grasp programming concepts.

Another 2024 study examined the problem-solving behavior of novice and experienced programmers using an automated assessment system. The results showed that experienced programmers made fewer errors and completed tasks more efficiently than beginners. The findings suggested that regular practice and continuous exposure to coding help develop stronger programming skills over time.

Papadakis et al. (2022) investigated formative assessment strategies in introductory programming courses. They found that frequent low-stakes assessments combined with timely feedback helped students recognize their weaknesses early and adjust their learning strategies. The study concluded that continuous feedback plays an important role in improving programming performance.

Luxton-Reilly et al. (2023) conducted a review of assessment practices in introductory programming education. Their findings showed that assessments focusing on the problem-solving process, rather than just final answers, provide a more accurate reflection of student learning. The study recommended designing assessments that match the real-world demands of programming tasks.

Pettit et al. (2022) studied the effects of automated feedback systems on programming learning outcomes. They found that students who received immediate feedback from these systems improved faster and produced more accurate code. The study highlighted that timely and specific feedback helps accelerate the development of programming skills.

Challenges in Programming Education

Ahadi et al. (2022) looked into the common struggles experienced by beginner programmers in introductory computer science courses. Their study found that many students had issues such as inconsistent practice, difficulty understanding programming syntax, and challenges in converting problem statements into working code. The researchers suggested that teachers should provide step-by-step guided activities that slowly build students' confidence and programming skills.

Robins (2022) revisited the persistent challenges in learning programming and examined how modern teaching methods have tried to address them. The study showed that although online tools and learning resources have made programming more accessible, students who do not practice regularly still find it hard to solve coding problems. The research emphasized that simply reading or watching tutorials is not enough—consistent hands-on practice is still the most important factor in learning programming.

Mhashi and Alakeel (2022) investigated why many students fail in programming courses at the university level. Their findings revealed that students who underestimated how much time and effort programming requires were more likely to perform poorly in problem-solving tasks. The study recommended that instructors and academic advisors help students plan realistic study schedules and develop better time management habits for programming subjects.

Koulouri, Lauria, and Macredie (2022) examined the challenges faced by first-generation university students in learning programming. The study found that limited prior experience, lack of sufficient practice time, and weak academic support often combined to make learning more difficult. Because of this, the researchers suggested the need for mentoring programs and targeted support systems to help students build strong foundational programming skills.

Research Gap

Although many recent studies have already explored programming practice, student engagement, cognitive development, and problem-solving skills, only a few have focused specifically on Bachelor of Science in Computer Science students in provincial state universities such as Surigao del Norte State University. Most of the available research has been conducted in international settings, larger institutions, or universities with more advanced technological resources. Because of this, the findings may not fully represent the actual situation and challenges

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

experienced by students in local schools. Due to this gap, the researchers believe that conducting a localized study is important and timely in order to better understand the relationship between programming practice hours and problem-solving skills among Computer Science students in the local context. The results of such a study may also be more applicable to institutions that share similar conditions and limitations.

III. RESEARCH METHODOLOGY

Research Design

This study used a quantitative descriptive-correlational research design to examine the relationship between programming practice hours and problem-solving skills among Computer Science students. This approach was chosen because it allows both variables to be measured in numerical form and analyzed in a systematic way to determine if a significant relationship exists between them. The descriptive part of the design helped the researchers describe the current levels of programming practice and problem-solving skills among the respondents, while the correlational part was used to identify the strength and direction of the relationship between the two variables..

Participants and Sampling Method

The participants of this study were 54 Computer Science students from Surigao del Norte State University. The respondents were chosen through convenience sampling, based on their availability and willingness to take part in the study. To be included, participants needed to be currently enrolled Computer Science students with at least basic experience in programming, ensuring that their responses were based on actual exposure to programming subjects.

Out of the 54 respondents, 27 were male and 27 were female, resulting in an equal distribution of gender. In terms of year level, most of the participants were first-year students, while the rest came from higher year levels. This mix of respondents provided a fairly balanced representation of students at different stages of their programming studies.

Data Collection Methods

The data in this study were collected using a structured questionnaire that was distributed to the respondents through Google Forms, making the process more convenient and efficient. The questionnaire was made up of 24 items designed to measure the two main variables of the study: Programming Practice Hours and Problem-Solving Skills.

The Programming Practice Hours section included indicators such as the number of hours spent coding each week, consistency of practice, frequency of programming activities, and participation in algorithm-related exercises. Meanwhile, the Problem-Solving Skills section focused on aspects such as analytical thinking, logical reasoning, identifying problems, and developing solutions. These indicators were carefully chosen to ensure that the questionnaire would properly reflect the key aspects of both variables.

The questionnaire utilized a 4-point Likert scale with the following values:

Scale	Description
4	Strongly Agree
3	Agree
2	Disagree
1	Strongly Disagree

Before the questionnaire was distributed, the respondents were informed about the purpose and nature of the study. Their participation was completely voluntary, and they were free to decline or withdraw at any time. All information collected from the respondents was handled with strict confidentiality and was used only for academic purposes. No personal identifying details were collected or disclosed throughout the data gathering process.

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

Data Analysis Techniques

The data collected from the questionnaire were analyzed using both descriptive and inferential statistics in order to fully address the objectives of the study.

Frequency and percentage were used to describe the respondents' demographic profile, particularly their gender and year level distribution. Meanwhile, the weighted mean was used to determine the overall level of programming practice hours and problem-solving skills among the participants, giving a clear numerical summary of each variable.

Based on the results, the overall mean for Programming Practice Hours was 2.97, while the overall mean for ProblemSolving Skills was 3.06. Both values fall under the "Agree" category of the Likert scale, which suggests that the respondents generally engage in programming practice at a moderate level and believe they have fairly developed problem-solving skills.

In addition, the Pearson Product-Moment Correlation was used to examine whether a significant relationship exists between programming practice hours and problem-solving skills. This method was appropriate since both variables were measured in a continuous form and the goal was to determine the direction and strength of their relationship. The results of this analysis became the basis for the conclusions and recommendations presented in the later chapters of the study.

IV. RESULTS AND DISCUSSION

This chapter presents the results obtained from the survey questionnaires completed by the respondents. The data were analyzed using frequency, percentage, and weighted mean to determine the level of programming practice habits and problemsolving skills of Computer Science students at Surigao del Norte State University. The findings are presented according to the respondents' demographic profile and their level of programming practice.

Profile of the Respondents

This part showed the demographic profile of the respondents in terms of age, sex, and year level.

Distribution by Age

Table 1. Age Distribution of Respondents

Age	Frequency	Percentage (%)
18 years old	16	29.63
19 years old	20	37.04
20 years old	9	16.67
21 years old	1	1.85
22 years old	3	5.56
23 years old	4	7.41
25 years old	1	1.85
Total	54	100.00

Table 1 presents the age distribution of the respondents. Most of them were 19 years old, with 20 students or 37.04% of the total population. This was followed by 18-year-old respondents, who accounted for 16 students or 29.63%. Those who were 20 years old made up 9 respondents or 16.67%, while 23-year-old students had 4 respondents or 7.41%. Respondents aged 22 years old consisted of 3 students or 5.56%. The least represented were those aged 21 and 25, with only 1 respondent each or 1.85%.

Overall, the results show that most of the respondents fall within the typical college age range. This suggests that the participants are mostly in the early stage of their college education, where they are still developing foundational knowledge in programming and technical skills. At this stage, students are generally adjusting to college life and beginning to build their coding habits. This period is important because the learning behaviors they develop early on can significantly influence their performance and skill development in the later years of their studies.

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

Distribution by Sex

Table 2. Sex Distribution of Respondents

Sex	Frequency	Percentage (%)
Male	27	50.00
Female	27	50.00
Total	54	100.00

Table 2 presents the sex distribution of the respondents. The results show that there was an equal number of male and female participants, with 27 students each, representing 50% of the total population.

This balanced distribution is important because it ensures that the findings are not influenced more by one group than the other. Both male and female respondents had equal opportunities to share their experiences and views regarding programming practice. It also reflects that students, regardless of sex, are actively participating in Computer Science programs and codingrelated activities. This balance suggests that involvement in technology-related fields is becoming more inclusive among students.

Distribution of Year Level

Table 3. Year Level Distribution of Respondents

Year Level	Frequency	Percentage (%)
1st Year	45	83.33
2nd Year	2	3.70
3rd Year	3	5.56
4th Year	4	7.41
Total	54	100.00

Table 3 presents the distribution of respondents according to their year level. Most of the respondents were first-year students, with 45 students or 83.33% of the total population. This was followed by fourth-year students with 4 respondents or 7.41%, and third-year students with 3 respondents or 5.56%. Second-year students had the lowest number, with only 2 respondents or 3.70%.

The results indicate that the majority of the participants were in their first year of college. As beginners, they were still in the process of learning basic programming concepts and adjusting to the demands of their course. Because of this, they are more likely to spend time practicing coding in order to better understand their lessons. The smaller number of respondents from higher year levels may be due to their heavier academic workload, including advanced subjects, research requirements, and other responsibilities that limited their availability to participate in the study.

Level of Coding Practice Habits

This section presents the level of coding practice habits among the respondents. The responses were analyzed using a weighted mean and interpreted using the following scale:

Scale	Verbal Interpretation
3.50 – 4.0	Very High
2.50 – 3.49	High
1.50 – 2.49	Low
1.00 – 1.49	Very Low

Table 4. Weighted Mean of Coding Practice Habits

DOI: 10.48175/IJARSCT-36575

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

Indicator	Weighted Mean	Verbal Interpretation
Weekly Practice Hours	2.97	High
Coding Consistency	2.97	High
Frequency of Practice	2.97	High
Algorithm Exercises	2.97	High
Overall Weighted Mean	2.97	High

Table 4 presents the weighted mean scores for each indicator of programming practice habits. All four indicators obtained the same weighted mean of 2.97, which was interpreted as "High." The overall weighted mean was also 2.97, indicating that the respondents generally demonstrate a high level of programming practice habits.

The results suggest that the students regularly allocate time for coding activities throughout the week. They also show consistency in practicing programming and often engage in algorithm-related exercises that help them develop logical thinking and problem-solving skills. These findings imply that programming practice has become a regular part of their academic routine.

The "High" rating further indicates that students recognize the importance of consistent practice in programming. Since programming is a skill that improves with continuous exposure and practice, students who frequently engage in coding are more likely to understand how programs work, identify and correct errors, and approach problems in a step-by-step manner. Regular practice also helps them become more comfortable and confident in handling programming tasks, which may contribute to better academic performance.

Interpretation of Findings

The findings of the study show that the respondents have a high level of programming practice habits, with an overall weighted mean of 2.97. This suggests that Computer Science students at Surigao del Norte State University are actively engaging in coding activities and are aware of the importance of regularly practicing programming.

It is also notable that most of the respondents were first-year students who, even at the beginning of their college journey, already showed active involvement in programming exercises. This is a positive indication that even beginners recognize the value of starting early in developing their coding skills. Early exposure gives students more time to strengthen their abilities, become more familiar with programming concepts, and develop effective study habits before encountering more advanced topics in higher years.

The equal number of male and female respondents also adds meaning to the results. It shows that both groups are equally engaged in programming practice, suggesting that interest in coding is not limited to a specific gender. This reflects a more inclusive trend in Computer Science education, where more students are becoming involved in programming regardless of gender.

Overall, the results suggest that consistent programming practice helps students better understand programming concepts, think more logically, and improve their problem-solving abilities. Students who regularly practice coding tend to feel more confident when writing programs, debugging errors, and working through coding challenges. These findings are consistent with previous studies, which emphasize that regular and continuous practice plays a key role in improving programming skills and developing computational thinking in Computer Science students.

Discussion of Unexpected Results

One of the noticeable results of the study was the uneven distribution of respondents across year levels. First-year students accounted for more than 80% of the total participants, while only a small number came from the second, third, and fourth years. This was not fully expected and may be due to the heavier workload of upper-year students, such as

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

more advanced subjects, research requirements, internships, and other academic responsibilities during the time the survey was conducted.

Another observation was that, although the respondents generally showed a high level of programming practice, some students still found it difficult to maintain consistent coding routines. A few of them appeared to struggle with managing their time, possibly because of a packed class schedule, limited access to resources, or difficulty balancing programming tasks with other academic requirements.

While these findings were somewhat unexpected, they provide a clearer understanding of the situation of Computer Science students in a local university setting. They also highlight the need for additional support from instructors and the institution, especially for upper-year students and those with limited resources, to help them stay consistent and engaged in programming practice throughout their studies.

V. CONCLUSIONS AND RECOMMENDATIONS

This chapter presents the conclusions and recommendations of the study on the relationship between programming practice hours and problem-solving skills among Computer Science students at Surigao del Norte State University. The conclusions are drawn from the responses of the 54 participants who took part in the study. The recommendations are also included for the benefit of students, teachers, school administrators, and future researchers.

Conclusions

The following conclusions were drawn based on the findings of the study:

Most of the respondents were 19 years old and were first-year college students. The sample also showed an equal number of male and female participants. This indicates that the respondents were mainly young, early-stage college students who are still in the process of learning the basics of programming.

The level of programming practice hours among the respondents was found to be high, with an overall weighted mean of 2.97. This suggests that the students regularly take part in coding activities such as weekly practice, algorithm exercises, and other consistent programming tasks. It also reflects that they recognize the importance of regular practice in improving their programming skills.

The level of problem-solving skills among the respondents was also rated high, with an overall weighted mean of 3.06. The students showed good performance in areas such as logical reasoning, analytical thinking, and developing solutions. However, problem identification was the weakest area among the indicators, suggesting that some students still need improvement in recognizing and clearly defining programming problems.

Overall, the study concluded that there is a positive relationship between programming practice hours and problem-solving skills. Students who spend more time practicing coding tend to perform better in problem-solving tasks. This indicates that regular exposure to programming activities plays an important role in helping students improve their ability to solve programming-related problems.

Recommendations

Based on the conclusions of the study, the following recommendations were offered:

For Students

Students are encouraged to develop consistent programming practice habits in order to further improve their problem-solving skills. Since regular practice has been shown to support analytical thinking and logical reasoning, students are advised to allocate enough time for coding-related activities. In particular, they are encouraged to:

1. Practice coding regularly through exercises, small projects, and algorithm-solving tasks to build familiarity and confidence in programming.
2. Improve their problem-identification skills by taking time to carefully read and fully understand programming problems before attempting to solve them.
3. Participate in coding challenges and group activities to strengthen their critical thinking and logical reasoning

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

abilities.

4. Maintain discipline and consistency in programming practice even outside of class hours, since skill improvement comes through continuous exposure and effort.

For Future and Instructors

Teachers and instructors are encouraged to provide more programming activities that can help strengthen students' problemsolving skills. They are advised to:

1. Design practical programming exercises that allow students to apply analytical and logical thinking in real coding situations.
2. Motivate students to practice coding independently while also providing guidance when they encounter difficult programming tasks.
3. Organize activities such as coding drills, debugging sessions, and problem-solving workshops to give students more hands-on experience.
4. Provide timely and constructive feedback on students' outputs so they can easily identify their mistakes and improve their programming performance.

For School

School administrators and curriculum developers are encouraged to support programs and initiatives that can further enhance students' programming practice and problem-solving skills. They are advised to:

1. Ensure that adequate laboratory facilities, stable internet connection, and necessary programming resources are available so students can practice coding effectively.
2. Organize seminars, workshops, and coding competitions that encourage the development of programming skills among Computer Science students.
3. Strengthen the programming curriculum by incorporating more hands-on activities, coding exercises, and problemsolving tasks that reflect real-world programming situations.
4. Support the use of modern learning tools and programming platforms that allow students to practice coding more efficiently and at their own pace.

For Future Researchers

Future researchers are encouraged to conduct related studies to further strengthen and validate the findings of this research. They are advised to:

1. Use a larger sample size that includes students from different colleges or universities to make the findings more representative and reliable.
2. Include other relevant variables such as academic performance, learning styles, motivation levels, and prior programming experience to provide a more complete understanding of factors affecting problem-solving skills.
3. Conduct comparative studies between different year levels or programs related to Information Technology and Computer Science to examine differences in programming practice habits and problem-solving skills across groups.
4. Explore other factors that may influence students' problem-solving skills aside from programming practice hours alone.
5. Use qualitative or mixed-method approaches to gain deeper insights into students' personal experiences and challenges in programming practice and problem-solving activities.
6. Examine more closely the relationship between academic performance and programming practice to better understand how they affect problem-solving skills.
7. Investigate the role of learning styles and motivation levels in improving students' programming performance.
8. Conduct deeper comparisons across different year levels to identify possible learning gaps in programming skills.
9. Explore other environmental or contextual factors that may affect students' ability to develop problem-solving skills

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

in programming.

VI ACKNOWLEDGEMENT

The proponents would like to give their deepest thanks to Surigao del Norte State University for the academic support and learning environment that made this study possible. The proponents are also truly grateful to their research adviser and panel members for the helpful guidance, useful feedback, and constant encouragement they gave throughout the entire research process. The participants who gave their time and answered the survey honestly are also warmly thanked, as their responses played a big part in making this study successful. The proponents would also like to thank their classmates and friends for the support and help they offered along the way, and their families for always being there with understanding, patience, and motivation during the whole duration of the study. Most of all, the proponents give thanks to God Almighty for the wisdom, strength, and determination He provided, which made it possible to finish this study.

REFERENCES

[1]. 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/10.1177/07356331221087773

[2]. Eckerdal, A., Berglund, A., & Thune, T. (2023). Learning programming practice and programming theory in the computer laboratory. European Journal of Engineering Education, 330–347. https://www.tandfonline.com/doi/full/10.1080/03043797.2023.2294953

[3]. Hwang, G. J., Tung, L. H., & Fang, J. W. (2023). Promoting students' programming logic and problem-solving awareness with precision feedback: A two-tier test-based online programming training approach. Journal of Educational Computing Research, 60(8), 1895–917. https://doi.org/10.1177/07356331221087773

[4]. Hsiao, T. C., Chuang, Y. H., Chen, T. L., Chang, C. Y., & Chen, C. C. (2022). Students' performances in computer programming of higher education for sustainable development: The effects of a peer-evaluation system. Frontiers in Psychology, 13, 911417. https://doi.org/10.3389/fpsyg.2022.911417

[5]. Prasad, A., Chaudhary, K., & Sharma, B. (2022). Programming skills: Visualization, interaction, home language and problem solving. Education and Information Technologies, 27(3), 3197–3223. https://doi.org/10.1007/s10639-021-10692-z

[6]. Hawa, D. M., Ghoniem, E., & Saad, A. M. (2022). Integrating problem-based learning into blended learning to enhance students' programming skills. Journal of Positive School Psychology, 6(8), 4479–4497. https://doi.org/10.53555/ks.v10i1.3554

[7]. Tavares, P. C., et al. (2022). Approaches to manage and understand student engagement in programming. Open Access, De Gruyter. https://www.researchgate.net/publication/359084528

[8]. Von Hausswolff, K. (2022). Student experiences in a university preparatory programming course. Frontiers in Computer

[9]. Science. https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2022.983237/full

[10]. Satratzemi, M., et al. (2023). Pair programming and oral exams for student engagement in programming courses. Computer Science Education. https://www.tandfonline.com/doi/pdf/10.1080/08993408.2024.2344401

[11]. Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). The effect of generative 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

[12]. Papadopoulou, F., Tsihouridis, C., & Batsila, M. (2022). Exploring the correlations between the dimensions of computational thinking and problem-solving concepts through students' perspectives. Mobility for Smart Cities and Regional Development. https://doi.org/10.1007/978-3-030-93904-5_67

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

[13]. Zhan, Z., He, W., Yi, X., & Ma, S. (2022). Effect of unplugged programming teaching aids on children's computational thinking and classroom interaction. Journal of Educational Computing Research, 60(5), 1277–1300. https://doi.org/10.1177/07356331211057143

[14]. De La Hoz, E., & Hijón, R. (2022). The impact of computational thinking on logical-mathematical reasoning in high school education: A quasi-experimental study. Education Sciences, 16(2), 345. https://www.mdpi.com/2227-7102/16/2/345

[15]. Weng, X., Ye, H., Dai, Y., & Ng, O. L. (2024). Integrating artificial intelligence and computational thinking in educational contexts: A systematic review of instructional design and student learning outcomes. Journal of Educational Computing Research. https://journals.sagepub.com/doi/10.1177/07356331241248686

[16]. Frontiers in Computer Science. (2024). Computational thinking in STEM education: Current state-of-the-art and future research directions. Frontiers in Computer Science. https://www.frontiersin.org/journals/computerscience/articles/10.3389/fcomp.2024.1480404/full

[17]. Abidin, A. F. Z., et al. (2025). Evaluating the difficulties in programming learning: Insights from polytechnic students. International Journal of Education and Practice, 14(1). https://hrmars.com/papers_submitted/24518/evaluating-thedifficulties-in-programming-learning-insights-from-polytechnic-students.pdf

[18]. Ericson, B. J., et al. (2022). Parsons problems and beyond: Systematic literature review and empirical study designs. Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education, 191–234. https://doi.org/10.1145/3571785.3574127

[19]. Denny, P., et al. (2022). Large language models for generating practice questions and personalized feedback in programming education. International Journal of STEM Education. https://link.springer.com/article/10.1186/s40594-025-00537-3