Enhancing Learning and Instruction through Structured Reflection in Pair Programming: A Feedback-Driven Approach in Computer Science Education

Dr. Oyku Eren Ozsoy, Embry-Riddle Aeronautical University - Prescott

Dr. Eren Ozsoy holds a Ph.D. in Health Informatics, along with an M.S. and B.S. in Computer Engineering. Her research interests include bioinformatics, machine learning, and linear optimization, with a specific focus on constructing biological networks and automating the classification of allergen proteins. Recently, Dr. Eren Ozsoy has also developed an interest in engineering education research and published an ASEE conference paper last year on the effects of ChatGPT on student learning in programming courses. With over seven years of experience teaching Computer Science courses, she is currently a faculty member at Embry-Riddle Aeronautical University's Department of Computer, Electrical, and Software Engineering, where she teaches computer science courses.

Dr. Luis Felipe Zapata-Rivera, Embry-Riddle Aeronautical University

Dr. Luis Felipe Zapata-Rivera is an Assistant Professor at Embry Riddle Aeronautical University. He earned a Ph.D. in Computer Engineering at Florida Atlantic University, in the past worked as an assistant researcher in the group of educational Technologies at Eafit University in Medellin, Colombia. His research area is the online Laboratories

Enhancing Learning and Instruction through Structured Reflection in Pair Programming: A Feedback-Driven Approach in Computer Science Education

Abstract

This study examines the potential impact of a structured reflection mechanism following weekly pair programming sessions on student learning, self-awareness, and skill development in a Computer Science course. The motivation for this research stems from the need to enhance students' critical thinking and ownership of their learning processes while helping them identify areas for improvement in technical skills such as syntax, loops, and functions. Additionally, it addresses the need for improvements in course design, instructional effectiveness, and the instructor's professional growth. The study employs qualitative and quantitative data collection involving two-course sections with a diverse group of students, engaging in 75-minute pair programming sessions where they alternate roles as driver (who writes the code) and navigator (who reviews and guides). The C programming language is used to facilitate collaboration and real-world skill development. The unique aspect of this study is the structured reflection process applied after each pair programming session. Students were asked to answer three questions: (1) what they learned, (2) what areas they needed more practice, and (3) their overall feedback on the exercise. These reflections provided qualitative insights into common challenges, including understanding loops, syntax, and random number functions, along with noted progress in collaboration and problem-solving skills. The findings highlight trends in student learning and areas for further instructional focus, offering valuable direction for course adjustments and teaching strategies.

To measure broader impacts, the beginning and end-of-semester surveys assess student satisfaction, confidence, and engagement, tracking changes in students' self-perception regarding their coding abilities and comfort with collaboration. By reviewing student feedback weekly, the instructor can create a more dynamic learning environment that evolves based on real-time insights. Overall, this study demonstrates how structured feedback mechanisms in pair programming can inform instructional practices and potentially promote students' self-confidence, adaptability, collaborative skills, and continuous learning while supporting dynamic and effective instructional practices. It highlights the significance of a feedback-driven approach in shaping teaching practices and supporting student growth, offering valuable insights for continuous improvement in the learning process.

1 Introduction

Pair programming is a coding technique in which two students work on the same task at one computer designing and coding the same algorithm. The "driver" writes the code, and the "navigator" observes, reviews, and guides the work of the driver through suggestions and corrections. The two students alternate roles as drivers and navigator and collaborate in algorithm design and coding.

Pair programming is considered one of the agile software development methodologies originating from Extreme Programming [1] and has been used for decades in a professional setting. Later, this method was adapted to teach programming in higher education. It has been used to enhance the learning process for beginners while promoting the development of effective teamwork and collaboration abilities.

Research on pair programming in higher education has aimed to understand the perspectives of the students on this technique and evaluate its benefits on the learning outcomes and academic performance. Many studies show that students generally have a positive opinion toward pair programming, appreciating the collaboration and communication it promotes [2], and development of teamwork skills [3, 4] and enjoying the team-oriented aspect of pair programming [3, 5, 6, 7, 8, 9, 10]. Furthermore, those who program in pairs frequently report greater confidence in the accuracy of their solutions compared to working individually [5, 6, 11, 12, 13]. Pair programming has also been linked to reduced frustration among students [12, 14] and is considered as a valuable tool for collaborative learning. Many students believe they gain more from working with a partner than they would on their own [4, 15].

The effect of pair programming on academic performance and learning, however, shows mixed results in the literature. Although most studies report that students working in pairs achieve higher grades, are more likely to complete courses successfully, and have higher pass rates [6, 7, 16, 17, 18, 19], there also are studies showing that no significant difference in assignment grades between pairs and solo programmers are observed [6, 9, 16, 20, 21], suggesting that pair programming may not directly affect knowledge retention. Some students have even expressed a preference for working alone, feeling that they better understand their programs when not paired [14].

Research on the effect of pair programming on program quality and productivity demonstrates that programs developed in pairs are of higher quality, with fewer errors [5, 6, 7, 9, 19]. For simpler problems, it was found that pair programming resulted in fewer mistakes [22]. However, findings on productivity are less consistent. While some research shows that pairs complete tasks more quickly [9, 23, 24], others suggest that the time required by pairs and individuals is comparable [5, 14].

The method of pairing students has also been investigated in the literature. Some studies assigned pairs randomly [17, 22], while others matched pairs based on academic performance [5, 25]. Research suggests that pairing students with slightly different skill levels may be optimal [26, 27], although differences in programming experience can sometimes cause stress within pairs.

While previous research on pair programming has focused primarily on student outcomes, our study extends the discussion by integrating a structured reflection mechanism to enhance both

student learning and instructional practices. Recent studies have highlighted the multifaceted benefits of pair programming in educational settings. For instance, a comprehensive literature review by Salleh et al. [23] indicates that pair programming can lead to increased success rates in introductory courses, higher quality software, and improved student confidence in solutions. Similarly, research by Celepkolu and Boyer [28] analyzed student reflections and found that pair programming positively influences cognitive, affective, and social experiences, contributing to a deeper understanding of programming concepts. Our study corroborates these findings by demonstrating that structured reflection within pair programming not only enhances learning outcomes but also provides actionable insights for instructional improvement. However, our research diverges from some existing studies regarding the impact of pair programming on student performance and interest. For example, a study by Bowman et al. [29] found no significant effect of pair programming on students' course performance or confidence. In contrast, our findings suggest that incorporating structured reflection into pair programming sessions can lead to notable improvements in both areas. This discrepancy may be attributed to differences in study design, participant demographics, or the specific implementation of reflective practices. By integrating structured reflection, our approach addresses potential limitations in traditional pair programming methodologies and offers a nuanced perspective on its efficacy in computer science education. This reflective approach not only informs real-time course adjustments but also provides insights into how students perceive their own progress and engagement. Additionally, this study contributes to the broader conversation on pedagogical strategies in Computer Science education by demonstrating how iterative feedback loops can support both student growth and instructor development. The following sections detail the methodology, including data collection techniques, survey design, and the analytical framework used to assess the impact of structured reflections on learning outcomes.

2 Study Goals and Methodology

2.1 Study Goals

This study aims to investigate the impact of structured reflections on student learning, self-awareness, and collaboration within a pair programming setting. Specifically, we define our goals as follows:

1. Support student learning and personal growth: By analyzing weekly reflections, we identify common learning challenges (e.g., syntax errors, debugging difficulties) and implement targeted interventions, such as additional exercises or guided discussions.
2. Promote instructor development: The instructor systematically adapts teaching strategies based on recurring student feedback, refining instructional approaches and course design.
3. Enhance engagement and adaptability: Student responses inform adjustments to the pair programming framework, ensuring that students actively participate and develop key technical and interpersonal skills.

By defining these objectives upfront, we establish clear criteria against which study outcomes can be measured.

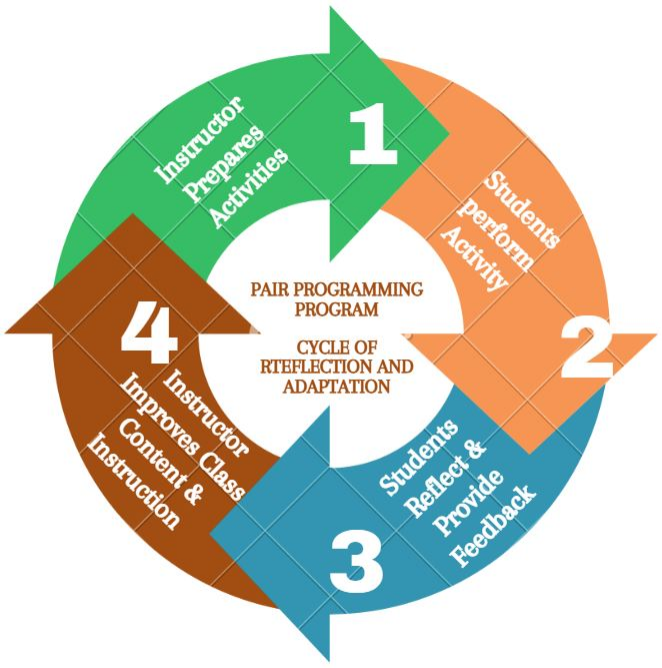

Figure 1: Iterative Cycle of Reflection and Adaptation

2.2 Study Design and Methodology

Building on these goals, this study employs a mixed-methods approach, combining qualitative and quantitative data collection. Two course sections participate in structured pair programming sessions, where students alternate between the roles of driver and navigator. Weekly reflections collected after each session address three core questions: (1) What did you learn? (2) What areas require more practice? (3) What feedback do you have on the experience? These responses undergo content analysis to identify trends, inform instructional improvements, and refine course materials.

2.3 Iterative Cycle of Reflection and Adaptation

After completing the activity in each pair programming session, students responded to three questions: i. what they learned or improved, ii. what they felt needed more practice, and iii. their thoughts or feedback about the exercise. These reflections encourage students to think more deeply about their learning, developing an awareness of their strengths and areas needing improvement while building habits to face new challenges. The responses of 42 students were reviewed using a content analysis approach. They were analyzed for recurring keywords, phrases, and concepts; grouped into categories based on their contextual meaning; and the frequency of each category was calculated. By reviewing weekly analyses, we developed action strategies to enhance student learning, used the insights to guide content improvements, and leveraged the results to better understand students' perspectives. This iterative process, illustrated in Fig. 1, aims to make the course more adaptable, creating a positive and supportive learning environment.

2.4 Action Strategies for Instructional Improvement

In response to weekly reflection data, the instructor implements targeted interventions, including:

1. Addressing Recurring Learning Challenges: When students repeatedly struggle with debugging, supplemental debugging tutorials and exercises are incorporated into lessons.
2. Enhancing Student Engagement: If feedback indicates that certain exercises are too complex or unengaging, adjustments are made to balance difficulty and interactivity.
3. Improving Instructor Awareness: Insights from reflections help the instructor identify gaps in explanations or areas needing reinforcement, leading to more effective lecture planning.

By explicitly linking reflections to actionable course modifications, this study demonstrates how structured feedback loops inform instructional improvements and enhance student learning outcomes.

2.5 Collaborative Inquiry of Pair Programming

Pair programming offers an engaging way for students to collaborate and learn actively. The instructor asks students to switch roles between the driver and the navigator every 10 minutes in a 75-minute class but allows flexibility if they need extra time to complete their current tasks. This allows each student to have a fair amount of time to experience both roles.

The instructor reviews feedback from students after each session, addressing challenges and supporting learning goals. This approach creates a partnership between students and the instructor, making the class more engaging, flexible, and effective.

2.6 Focus on Professional Growth

One of the most important aspects of this study is its benefit from regular feedback. By reviewing weekly structured reflections, the instructor spots common challenges (e.g., syntax or debugging) and successes. The feedback also highlights areas where students realize their strengths and weaknesses.

Each week, the instructor creates a summary to share with the class. This summary covers common issues, such as difficulty with functions or concerns about exercises being too long or repetitive. It also includes unique and meaningful comments, such as; “I feel I need to express myself better in group work.” These comments are shared anonymously with the class, sparking class discussions. This approach helps students see how their input shapes the class and supports their learning.

The feedback loop allows the instructor to adjust lessons, activities, and materials to better match student needs, making the class more engaging and flexible. At the same time, reflecting on feedback helps the instructor refine their teaching strategies and make informed decisions. By acting on real-time insights, the instructor builds a deeper understanding of how students learn and creates a more targeted and effective teaching approach.

2.7 Research Questions and Hypothesis

The design of the questionnaire was guided by the following research question and hypotheses.

Research question:

"How do weekly structured reflections after pair programming sessions impact students' confidence in communication, self-awareness, and teamwork skills?"

Hypotheses:

1) The pair programming program does not cultivate students' teamwork skills by promoting collaboration and shared problem-solving (H0). The pair programming program cultivates students' teamwork skills by promoting collaboration and shared problem-solving (H1).

2) Weekly structured reflections do not empower students to build confidence in effectively communicating with faculty members and peers (H0). Weekly structured reflections empower students to build confidence in effectively communicating with faculty members and peers (H1).

2.8 Participants

For the study, two sections of the CS 125 Computer Science I course offered in Fall 2024 semester were selected as the target population. This course, integral to the engineering curriculum, teaches C programming language as a core component.

The required sample size for the study was determined using the standard sampling calculations described below with a confidence level of 95% and a margin of error of 5%:

1) For an infinite population, the sample size S can be calculated by using the equation:

$$S = Z^2 * P * (1 - P)/M^2, \tag{1}$$

where *Z* is the z-score based on the desired confidence level, *P* is the estimated population proportion (often assumed as 0.5), and *M* is the desired margin of error. For this survey, the sample size S is calculated as:

$$S = 1.96^2 * 0.5 * (1 - 0.5)/0.05^2 = 384.16, \tag{2}$$

2) The value of S should be adjusted for a finite population by using the equation:

$$A = n/(1 + (n–1)/P), \tag{3}$$

where *A* is the adjusted sample size based on the population n and *P* is the total population in this category.

$$A = 384.16/(1 + (384.16 - 1)/42) = 37.95. \quad (4)$$

This calculation suggests that with 38 participants that submit responses for the surveys, the study will have 95% confidence with a margin of error of 5%.

Accordingly, we selected a total population of 42 students in this category, encompassing the following course sections:

Table 1: Participants Description

Class Name	Section ID	Level	# of Students
CS 125 Computer Science I	01PC	Freshman, Sophomore, Junior	23
CS 125 Computer Science I	02PC	Freshman, Sophomore, Junior	19
		Total	42

Given the study's intent to explore broader student learning trends, this sample size may limit the generalizability of findings. As a result, the conclusions drawn should be interpreted as insights specific to this cohort rather than definitive trends for all students. Future studies with a larger sample size would be beneficial for confirming these patterns on a broader scale.

2.9 Data Collection

The study employs a mixed-methods approach, combining qualitative data from weekly structured reflections and quantitative data from comprehensive surveys. Together, these methods provide a detailed understanding of students' learning progress, confidence, and collaboration skills, as well as insights into instructional effectiveness. Qualitative data are gathered through open-ended reflections completed weekly by students after pair programming sessions. These reflections prompt students to articulate what they learned, identify areas where they need more practice, and share their overall experience, including feedback on collaboration with their partners. This qualitative input captures students' real-time challenges, progress, and perceptions, offering insights into learning patterns and collaboration dynamics. It also allows the instructor to address recurring issues such as difficulties with understanding loops, syntax, and debugging while making iterative adjustments to improve instruction. Quantitative data are collected through a survey. The survey consists of questions across multiple dimensions to track changes in skills, confidence, and the impact of the reflection process. In addition to numerical ratings, the survey includes open-ended questions that allow students to provide detailed examples, explanations, and insights into their learning experiences.

Survey Design

The designed survey aims to assess the impact of pair programming and reflection practices on student learning, confidence, and collaboration. It comprises a mix of Likert scale questions (1-5 scale: 1 lowest and 5 highest) and open-ended reflection-driven queries to capture both quantitative and qualitative data about the students' experiences. The survey consists of three sections: one section with two overarching reflection questions, and the other two sections

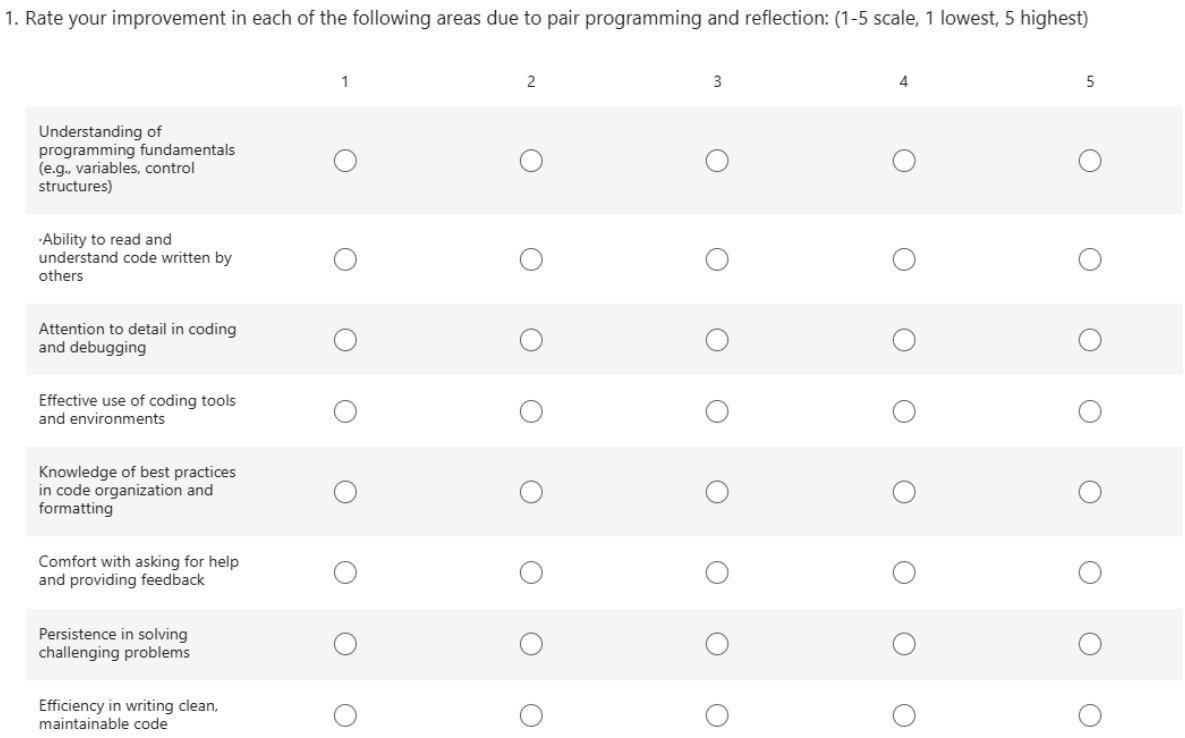

Figure 2: Skill improvement question used in the Survey

containing a total of 15 specific questions. Each section is designed to gather specific insights and track students' progress throughout the semester.

Section 1 - Overarching Reflection Questions:
Section 1 includes only the two overarching reflection questions. In the first question of this section, students rate their improvement in key areas such as programming fundamentals, debugging, problem-solving, and collaboration using a 1-5 scale. This question from the survey is presented in Fig. 2.

The second question is given in Fig. 3. This question identifies the specific skills that students feel have improved the most through their experiences with pair programming and reflections. The question format allows students to select multiple areas, such as problem-solving, debugging, communication, and adaptability to new coding challenges.

These two questions provide a high-level perspective on students' growth over the semester.

Section 2 - General Pair Programming Questions:
Section 2 comprises general pair programming questions, covering Question 3 to Question 10. This section evaluates the impact of pair programming sessions on students' problem-solving abilities, understanding of programming concepts, and confidence in coding. By collecting data at two points in time, the beginning and the end of the semester, this section enables the analysis of changes and improvements over the semester.

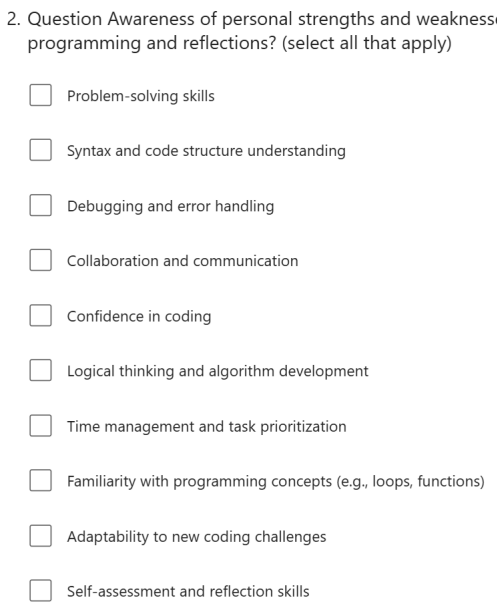

Figure 3: Survey Question on Skills Improved Through Pair Programming and Reflections

Section 3 - Feedback and Reflection Questions:
Section 3 focuses on questions related to feedback and reflections, covering Question 11 to Question 17. It is designed to assess the impact of reflection and feedback-driven sessions, as well as the effectiveness of engaging with reflection questions throughout the semester.

Common Features of Sections 2 and 3:
Sections 2 and 3 include question groups, each consisting of one Likert scale question and one open-ended question. The Likert scale questions track progress by asking about key skills or confidence levels. The open-ended questions allow students to express their thoughts in their own words, adding a personal touch and providing examples that bring the data to life, helping to understand the ”why” behind the numbers.

These sections evaluate how feedback sessions shaped students’ confidence, their ability to take control of their learning, and their comfort with self-expression. By asking questions at the start and end of the semester, the survey enables an analysis of how these areas may have evolved over time.

All of these questions aim to provide a thoughtful perspective on both the outcomes and the experiences that influenced them.

3 Results and Discussion

The weekly structured reflection questions were collected for 10 weeks during the semester and they were completed by 42 students (100 % of the students participating in the study). On the other hand, the beginning-semester and end-of-semester surveys were completed by 40 students (95.2% of the students participating in the study). The content analysis of weekly structured

reflections and the analytical statistics of survey results are presented below.

3.1 Content Analysis of Weekly Structured Reflections

In this section, we provide a detailed analysis of the first week's responses as an example to highlight our methodology and findings. The analyses for subsequent weeks are included in the Appendix for reference.

Table 2: Content Analysis of First Pair Programming Reflections

Category	Frequency	Example from Responses
Syntax and Formatting	28	"I need more practice remembering to use `&` in `scanf` statements." "I often forget semicolons, which leads to errors."
Conditional Logic	20	"I learned how to use if-else statements more effectively." "I improved on nested conditionals and decision trees."
Collaboration	18	"Working with a partner made debugging easier." "I appreciated the teamwork aspect of the exercise."
Debugging	15	"I learned to debug using progressive testing." "I need to improve error checking techniques."
Engagement and Fun	12	"I loved today's exercise because it was fun and interactive." "The thematic exercises like potion shops are enjoyable."
Tool Usage	8	"I relearned how to use MobaXterm to manage files." "Separating code into files was a helpful organizational practice."

Insights from Content Analysis of First Week's Structured Reflections and Actionable Strategies:

Feedback collected highlights several key areas where students experienced growth, faced challenges, and found value in the exercises. Approximately 67% of the students (28 out of 42) mentioned syntax-related issues, highlighting the need for targeted exercises on semicolons, format specifiers (%d, %f), and the correct usage of scanf. About 48% (20 out of 42) of responses emphasized improvements or challenges related to if-else statements, nested conditionals, and logical operators. Many students appreciated the peer programming model; emphasizing how it improved communication and debugging. More than a third of the students mentioned struggles with debugging or error handling, indicating a need for exercises that simulate real-world debugging scenarios. The students explicitly mentioned enjoying thematic exercises. According to the analysis, the syntax and debugging categories were associated with predominantly negative sentiments, reflecting frustration or difficulty in these areas. On the other hand, collaboration and thematic exercises received positive sentiments, highlighting their value in peer-supported learning.

Based on these findings, we identify recurring difficulties, such as syntax errors and debugging, and recognize the positive impact of collaborative and thematic exercises on student engagement. Our group has created actionable strategies to address the challenges identified while reinforcing the successful aspects of the learning environment. For the first week, we focused on quick exercises to tackle common syntax problems like semicolon placement and scanf usage and decided to add guided exercises that mimic real-world debugging scenarios, teaching students to identify and resolve errors.

3.2 Analytical Statistics of Survey Results

In this section, we evaluate the hypotheses mentioned in the previous section to determine whether there is a significant difference in students' confidence levels between the beginning and the end of the semester.

Open-ended responses were systematically reviewed and categorized into themes based on recurring patterns observed in the data. The responses were analyzed for common keywords, phrases, and concepts, which were grouped into categories such as syntax challenges, debugging strategies, collaboration, and engagement. Both authors independently reviewed the responses and discussed discrepancies to refine the theme classification, ensuring consistency and minimizing potential bias. The inclusion of representative quotes in this section is based on their alignment with dominant themes rather than selective reporting.

Questions 1 and 2 of the survey asked the students to report their perception of improvement and their awareness of the level of knowledge on different topics. The average scores for these questions are presented in Figs. 4 and 5.

Table 3: Intermediate results of the Statistical Analysis for Survey Section-2

Question #	Pre	Post	Diff (Post - Pre)	Dev (Diff - M)	Sq. Dev
3	2.75	4.22	1.47	0.31	0.1
4	2.76	4.25	1.48	0.31	0.1
5	2.8	4.17	1.38	0.21	0.04
6	3.3	4.12	0.83	-0.34	0.12
7	3.4	4.17	0.77	-0.39	0.15
8	3	4.22	1.22	0.06	0
9	2.67	4.07	1.4	0.23	0.05
10	3.27	4.05	0.77	-0.39	0.15
			M: 1.17		SS: 0.72
M: mean of the difference between the two surveys (Post-Pre) SS: sum of squares of deviations					

3.2.1 Hypotheses Testing

Hypothesis 1) The pair programming program does not cultivate students' teamwork skills by promoting collaboration and shared problem-solving (H0). The pair programming program

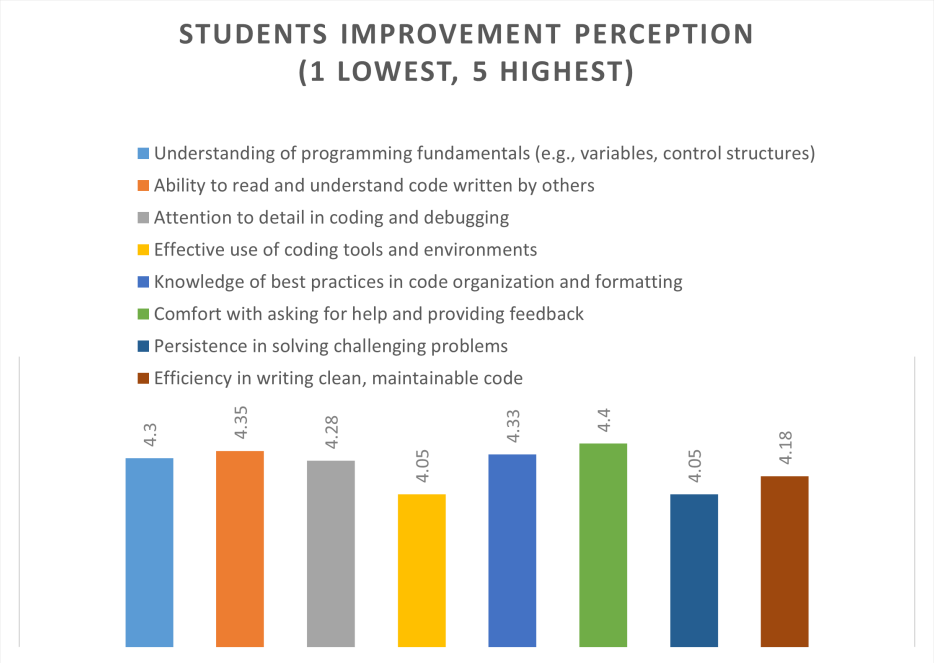

Figure 4: Survey Question-1 : Rate your improvement in each of the following areas due to pair programming and reflection: (1-5 scale, 1 lowest, 5 highest)

cultivates students' teamwork skills by promoting collaboration and shared problem-solving (H1).

The means of the responses collected from participants at the beginning (Pre) and the end (Post) of the semester were calculated for each of the eight questions in Section-2 of the survey, and they were used for statistical analysis. The intermediate results for the calculation of statistical metrics used for testing Hypothesis-1 are presented in Table 3. Considering a mean population of μ = 0, and a degree of freedom *dof* = 8-1 = 7, the t-value is calculated by using:

$$t = (M - \mu)/(S/\sqrt{N}) \text{ where } S = \sqrt{SS/dof} \tag{5}$$

and therefore;

$$S = \sqrt{SS/dof} = \sqrt{0.72/(8-1)} = 0.32 \text{ and } t = (1.17 - 0)/(0.32/\sqrt{8}) = 10.3 \tag{6}$$

The two-tailed p-value corresponding to a *t*-value of $t = 10.3$ is $p = 0.00002$. The result is significant at $p < 0.05$, which means that for this set of values, there is a difference between the pre- and post-test results.

Hypothesis 2) Weekly structured reflections do not empower students to build confidence in effectively communicating with faculty members and peers (H0). Weekly structured reflections empower students to build confidence in effectively communicating with faculty members and peers (H1).

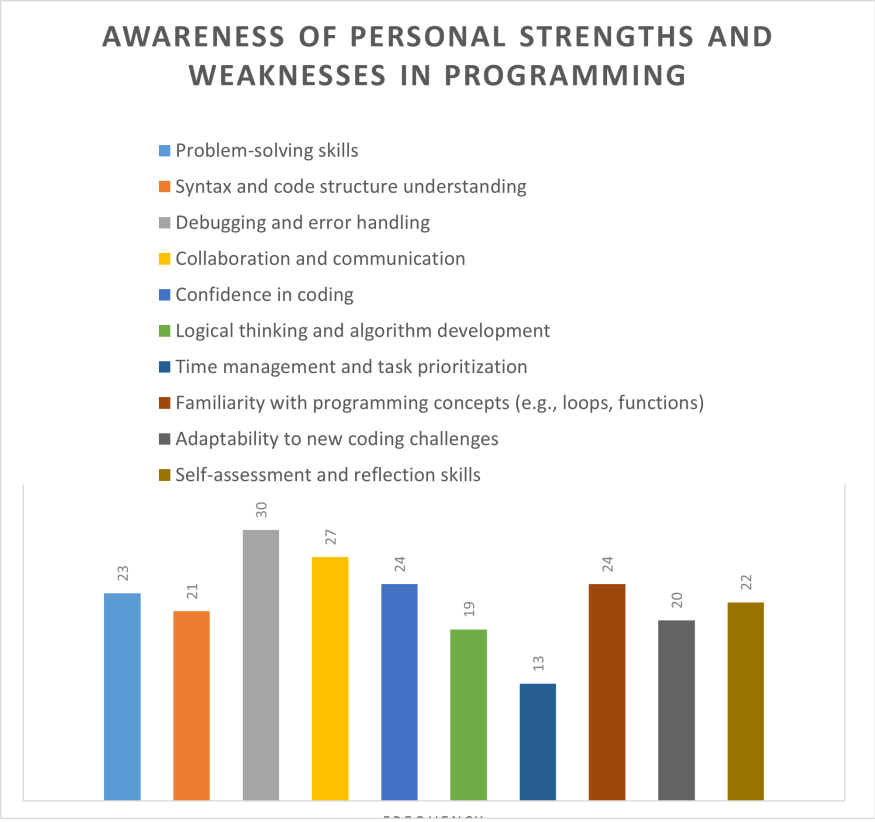

Figure 5: Survey Question-2 : Which skills do you feel have most improved from pair programming and reflections? (select all that apply)

Applying the same statistical method used for Hypothesis 1 to the seven questions in Section-3 of the survey, the intermediate results of which is presented in Table 4, Eqn. 5 calculates:

$$S = \sqrt{SS/dof} = \sqrt{0.44/(7-1)} = 0.27 \text{ and } t = (1.02-0)/(0.27/\sqrt{7}) = 10.01 \quad (7)$$

The two-tailed p-value corresponding to a *t*-value of $t = 10.01$ is $p = 0.00006$. The result is significant at $p < 0.05$, which means that for this set of values, there is a difference between the pre- and post-test results.

Table 4: Intermediate results of the Statistical Analysis for Survey Section-3 and 4

Question #	Pre	Post	Diff (Post - Pre)	Dev (Diff - M)	Sq. Dev
11	2.95	3.92	0.97	-0.05	0
12	3.22	4.4	1.18	0.15	0.02
13	3.27	4.3	1.02	0	0
14	3.2	3.95	0.75	-0.27	0.08
15	3.3	4.12	0.83	-0.2	0.04
16	2.7	4.25	1.55	0.53	0.28
17	3.15	4.02	0.87	-0.15	0.02
			M: 1.02		SS: 0.44
M: mean of the difference between the two surveys (Post-Pre) SS: sum of squares of deviations					

3.2.2 Descriptive Results of Survey Questions

This section provides a detailed descriptive results of the survey questions.

Q3 - Self-Efficacy in Problem Solving: Rate your confidence in solving programming problems independently.
Students' confidence in solving programming problems independently grew significantly:

- Pre-course average confidence: 2.75/5.
- Post-course average confidence: 4.22/5.
- Students credited feedback from partners and structured reflection for this improvement, with one participant stating:

  > "Pair programming gave me new strategies for breaking down complex problems, which made me feel more capable."

Q4 - Understanding Core Programming Concepts: Rate your understanding of core programming concepts.
Students reported understanding of core programming concepts improved over the course:

- Pre-course understanding: 2.76/5.
- Post-course understanding: 4.25/5.

- Discussions with partners were rated as highly effective in clarifying these concepts, with an average impact score of 4.3/5.

Q5 - Adaptability to New Programming Challenges: Rate your comfort level with tackling unfamiliar programming challenges.
Comfort in tackling unfamiliar programming tasks improved:

- Pre-course comfort level: 2.8/5.
- Post-course comfort level: 4.17/5.
- Students highlighted that pair programming helped develop strategies for new challenges, with one commenting:

  > "Having a partner to brainstorm with made unfamiliar problems feel less intimidating."

Q6 - Ability to Give and Receive Feedback: Rate your confidence in giving constructive feedback.
Students reflected their ability to give and receive feedback increased:

- Pre-course constructive feedback level: 3.3/5.
- Post-course constructive feedback level: 4.12/5.
- One student shared:

  > "Learning to give detailed feedback improved my ability to articulate my thought process and collaborate effectively."

Q7 - Engagement and Motivation in Programming: Rate your motivation to engage with programming tasks.
Students reported engagement with programming tasks was positively influenced by pair programming:

- Motivation pre-course: 3.4/5.
- Motivation post-course: 4.17/5.
- Students described the collaborative aspect as highly motivating, with one stating:

  > "Working with a partner made every task feel like a shared challenge, which kept me engaged."

Q8 - Reflection on Strengths and Weaknesses: Rate your awareness of your strengths and weaknesses in programming.
Students reflected increase in awareness of their strengths and weaknesses:

- Pre-course awareness: 3.0/5.
- Post-course awareness: 4.22/5.
- One student noted:

> "I realized that my biggest weakness is spelling and syntax errors, especially with pointers, but working through exercises helped me improve."

- Another student shared:

> "My most significant strength was collaborating with my peers during pair programming, which helped me learn faster and tackle more complex challenges."

Q9 - Perceived Improvement in Programming Efficiency: Rate your efficiency in completing programming tasks.
Efficiency in completing programming tasks showed marked improvement:

- Pre-course efficiency: 2.67/5.
- Post-course efficiency: 4.07/5.
- One student shared:

> "Switching roles helped my efficiency. Having that 10-minute marker helped me think and write my code more efficiently."

- Another student noted:

> "Pair programming gave me opportunities to see how other people go about solving and formatting their code for different problems."

Q10 - Collaboration Willingness in Programming Projects: Rate your confidence in collaborating with others on programming projects.
Students reported increased confidence and skills in collaborating on programming projects:

- Pre-course collaboration level: 3.27/5.
- Post-course collaboration level: 4.05/5.
- One student noted:

> "Pair programming taught me how to communicate my ideas and listen to others, which has been invaluable in team settings."

- Another student reflected:

> "I learned that teamwork can be great for things like projects, as it allows you to learn from others and approach problems with fresh perspectives."

Q11 - Enhancement of Learning Experience: Rate your confidence in engaging with structured feedback mechanisms.
Students reflected on how structured feedback helped improve their understanding and approach:

- Pre-course confidence in engaging with structured feedback mechanisms level: 2.95/5.
- Post-course confidence in engaging with structured feedback mechanisms level: 3.92/5.

- One student shared:

  > ”Feedback from these sessions have all really helped me understand where I might have issues or where I’m not fully grasping the concept. It helps to solidify what I’m learning.”

Q12 - Self-Expression with the Professor: Rate your confidence in expressing your thoughts and concerns to professors.
Students reflected on how their confidence in expressing thoughts and concerns to professors improved throughout the course:

- Pre-course confidence in expressing their thoughts and concerns to professors level: 3.22/5.
- Post-course confidence in expressing their thoughts and concerns to professors level: 4.4/5.
- One student shared:

  > ”When the professor changed things in the course to make it work better for everyone it made me feel like I was heard.”

- Another student noted:

  > ”The professor creates a document by using our feedback after each section and explains what is the plan based on our feedback.”

Q13 - Perception of Being Seen and Understood: Rate how much you felt your feedback was valued.
Students reflected on how their feedback was valued and how it influenced their learning experience:

- Pre-course confidence in feedback being valued level: 3.27/5.
- Post-course confidence in feedback being valued level: 4.3/5.
- One student shared:

  > ”Being able to give feedback made me feel heard and understood since I knew that there were changes being made to the class to help cater to my needs.”

- Another student noted:

  > ”I really liked the weekly feedback surveys, because I could see a change in the assignments over the weeks of the course in the direction of the feedback offered.”

- Another student remarked:

  > ”The professor helps tailor the class to the needs of the students that semester, and I felt my feedback was taken into heavy consideration.”

Q14 - Advocacy Skills in Other Classes: Rate how confident you are in advocating for yourself in other classes.
Students reflected on how their advocacy skills developed and their ability to express their needs and concerns in other academic settings:

- Pre-course confidence in advocating for themselves in other classes: 3.2/5.
- Post-course confidence in advocating for themselves in other classes: 3.95/5.
- One student shared:

  > ”Feedback in this class has helped me see how voicing my side of things in other classes generally helps improve my learning and understanding of a concept.”

- Another student noted:

  > ”Feedback practices in this class made me more comfortable with giving feedback to other professors. This has taught me how to express my concerns in a respectful manner.”

Q15 - Effect of Identifying Learned Skills or Improvements: Rate your ability to recognize what you learned or improved.
Students reflected on how their ability to identify learned skills or areas of improvement developed over the course:

- Pre-course Identifying Learned Skills or Improvements level: 3.3/5.
- Post-course Identifying Learned Skills or Improvements level: 4.12/5.
- One student shared:

  > ”This has definitely helped my mind to realize what I was working on and think about it more. This was especially apparent with arrays. When doing the exercise I wasn’t thinking much but I thought more after this reflection.”

- Another student noted:

  > ”I wouldn’t say reflection helped me better understand concepts, but rather help me understand what areas I needed to improve in.”

Q16 - Effect of Identifying Areas for Practice: Rate how aware you are of the areas in programming where you need more practice.
Students reflected on how identifying areas for practice improved over the course:

- Pre-course Identifying Areas for Practice level: 2.7/5.
- Post-course Identifying Areas for Practice level: 4.25/5.
- One student shared:

  > ”This question made me go back into the code and really think about what I struggled on and what questions I had. It also provided an environment to ask any lingering questions I had one on one versus the class environment.”

- Another student noted:

  > ”Identifying my struggles and reflecting on them taught me and made me realize the importance of what I did and how to do it better.”

Q17 - General Impact of Reflection Questions: Rate your ability to engage in self-reflection about your learning and practice.
Students reflected on how the reflection questions influenced their approach to programming and problem-solving:

- Pre-course level: 3.15/5.
- Post-course level: 4.02/5.
- One student shared:

  > ”The reflection questions helped me understand what concepts I need to work on and focus on areas where I did well to apply it to the next problem.”

- Another student remarked:

  > ”I can now analyze my strengths and weaknesses better for the final and apply that knowledge to improve my work.”

4 Conclusion

This study demonstrates the potential benefits of integrating structured reflection after pair programming sessions to enhance student learning in a Computer Science course. By incorporating weekly structured reflections, students were prompted to assess their progress, identify areas for improvement, and evaluate their collaboration skills. The recurring nature of these reflections allowed students to develop a deeper sense of self-awareness and adaptive learning, as they continuously engaged in critical thinking about their coding practices.

Based on the students comments, one of the key aspects identified in this study was the level of awareness about their improvement in technical abilities, and also their sense of ownership over their learning, enhancing their confidence in their problem-solving and collaboration skills.

In addition, the study emphasizes how the feedback mechanism benefited the instructor’s professional development and course design. By reviewing student reflections weekly, the instructor was able to identify trends and challenges in real-time, which informed adjustments to the course structure and instructional strategies. This feedback-driven approach allowed for a dynamic learning environment that evolved to meet students’ needs more effectively. The use of surveys conducted at the beginning and end of the semester further highlighted the broader impact of the reflective process, capturing students’ growing satisfaction, confidence, and comfort with collaboration. Ultimately, the study suggests that structured reflection in pair programming not only aids students in their technical and collaborative skill development but also supports instructors in creating more responsive and engaging learning environments.

Future research will involve conducting a study with both a control group and an experimental group to assess the true impact of implementing a pair programming model combined with weekly structured reflections on student learning outcomes. This approach will help determine the effectiveness of this strategy in enhancing students’ skills and overall learning experience.

References

[1] Kent Beck. *Extreme programming explained: embrace change*. Addison-Wesley Longman Publishing Co., Inc., USA, 1999. ISBN 0201616416.

[2] Elizabeth V. Howard. Attitudes on using pair-programming. *Journal of Educational Technology Systems*, 35(1): 89–103, 2006. doi: 10.2190/5K87-58W8-G07M-2811.

[3] Daniel C. Cliburn. Experiences with pair programming at a small college. *Journal of Computing Sciences in Colleges*, 19:20–29, 2003.

[4] Richard L. Edwards, Jennifer K. Stewart, and Mexhid Ferati. Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM Inroads*, 1:48–54, 2010.

[5] Laurie A. Williams and Robert R. Kessler. Experiments with industry's "pair-programming" model in the computer science classroom. *Computer Science Education*, 11(1):7–20, 2001. doi: 10.1076/csed.11.1.7.3846.

[6] Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald. Pair programming improves student retention, confidence, and program quality. *Commun. ACM*, 49(8):90–95, August 2006. ISSN 0001-0782. doi: 10.1145/1145287.1145293.

[7] Wallace Chigona and Michael Pollock. Pair programming for information systems students new to programming: Students' experiences and teachers' challenges. In *PICMET '08 - 2008 Portland International Conference on Management of Engineering Technology*, pages 1587–1594, 2008. doi: 10.1109/PICMET.2008.4599777.

[8] J. L. van der Walt E. Mentz and L. Goosen. The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education*, 18(4):247–260, 2008. doi: 10.1080/08993400802461396.

[9] Nick Z. Zacharis. Measuring the effects of virtual pair programming in an introductory programming java course. *IEEE Transactions on Education*, 54:168–170, 2011.

[10] Stelios Xinogalos, Maya Satratzemi, Alexander Chatzigeorgiou, and Despina Tsompanoudi. Student perceptions on the benefits and shortcomings of distributed pair programming assignments. In *2017 IEEE Global Engineering Education Conference (EDUCON)*, pages 1513–1521, 2017. doi: 10.1109/EDUCON.2017.7943050.

[11] Linda L. Werner, Brian Hanks, and Charlie McDowell. Pair-programming helps female computer science students. *J. Educ. Resour. Comput.*, 4(1):4–es, March 2004. ISSN 1531-4278. doi: 10.1145/1060071.1060075.

[12] Grant Braught, Tim Wahls, and L. Marlin Eby. The case for pair programming in the computer science classroom. *ACM Trans. Comput. Educ.*, 11(1), February 2011. doi: 10.1145/1921607.1921609.

[13] Xuefeng Wei, Lin Lin, Nanxi Meng, Wei Tan, Siu-Cheung Kong, and Kinshuk. The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. *Computers Education*, 160:104023, 2021. ISSN 0360-1315. doi: https://doi.org/10.1016/j.compedu.2020.104023.

[14] Beth Simon and Brian Hanks. First-year students' impressions of pair programming in cs1. *J. Educ. Resour. Comput.*, 7(4), January 2008. ISSN 1531-4278. doi: 10.1145/1316450.1316455.

[15] J C Carver, L Henderson, L He, J Hodges, and D Reese. Increased retention of early computer science and software engineering students using pair programming. Institute of Electrical and Electronics Engineers (IEEE), jul 2007.

[16] Kai Yang Miriam Ferzli Laurie Williams, Eric Wiebe and Carol Miller. In support of pair programming in the introductory computer science course. *Computer Science Education*, 12(3):197–212, 2002. doi: 10.1076/csed.12.3.197.8618.

[17] Emilia Mendes, Lubna Al-Fakhri, and Andrew Luxton-Reilly. A replicated experiment of pair-programming in a 2nd-year software development and design computer science course. *SIGCSE Bull.*, 38(3):108–112, June 2006. ISSN 0097-8418. doi: 10.1145/1140123.1140155.

[18] Karthikeyan Umapathy and Albert D. Ritzhaupt. A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Trans. Comput. Educ.*, 17(4), August 2017. doi: 10.1145/2996201.

[19] Renée McCauley Laurie Murphy Brian Hanks, Sue Fitzgerald and Carol Zander. Pair programming in education: a literature review. *Computer Science Education*, 21(2):135–173, 2011. doi: 10.1080/08993408.2011.579808.

[20] Baichang Zhong, Qiyun Wang, and Jie Chen. The impact of social factors on pair programming in a primary school. *Computers in Human Behavior*, 64:423–431, 2016. ISSN 0747-5632. doi: https://doi.org/10.1016/j.chb.2016.07.017.

[21] Fan Xu and Ana-Paula Correia. Adopting distributed pair programming as an effective team learning activity: a systematic review. *Journal of Computing in Higher Education*, 36(2):320–349, 2024. ISSN 1867-1233. doi: 10.1007/s12528-023-09356-3.

[22] Matthias M. Müller. Do programmer pairs make different mistakes than solo programmers? *Journal of Systems and Software*, 80(9):1460–1471, 2007. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2006.10.032. Evaluation and Assessment in Software Engineering.

[23] Norsaremah Salleh, Emilia Mendes, and John Grundy. Empirical studies of pair programming for cs/se teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering*, 37 (4):509–525, 2011. doi: 10.1109/TSE.2010.59.

[24] Herez Moise Kattan, Flavio Soares, Alfredo Goldman, Eduardo Deboni, and Eduardo Guerra. Swarm or pair? strengths and weaknesses of pair programming and mob programming. In *Proceedings of the 19th International Conference on Agile Software Development: Companion*, XP '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450364225. doi: 10.1145/3234152.3234169.

[25] Kyungsub Steve Choi, Fadi P. Deek, and Il Im. Pair dynamics in team collaboration. *Computers in Human Behavior*, 25(4):844–852, July 2009. ISSN 0747-5632. doi: 10.1016/j.chb.2008.09.005.

[26] Theodore Van Toll, Roger Y. Lee, and Thomas Ahlswede. Evaluating the usefulness of pair programming in a classroom setting. *6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)*, pages 302–308, 2007.

[27] J. Bevan, L. Werner, and C. McDowell. Guidelines for the use of pair programming in a freshman programming class. In *Proceedings of the 15th Conference on Software Engineering Education and Training*, CSEET '02, page 100, USA, 2002. IEEE Computer Society.

[28] Mehmet Celepkolu and Kristy Elizabeth Boyer. The importance of producing shared code through pair programming. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 765–770, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351034. doi: 10.1145/3159450.3159506.

[29] Nicholas A. Bowman, Lindsay Jarratt, KC Culver, and Alberto M. Segre. The impact of pair programming on college students' interest, perceptions, and achievement in computer science. *ACM Trans. Comput. Educ.*, 21 (3), May 2021. doi: 10.1145/3440759.

APPENDIX:

A1. Survey Questions

Q1: Rate your improvement in each of the following areas due to pair programming and reflection (1-5 scale, 1 lowest, 5 highest):

Skill/Area of Improvement	**1**	**2**	**3**	**4**	**5**
Understanding of programming fundamentals (e.g., variables, control structures)	○	○	○	○	○
Ability to read and understand code written by others	○	○	○	○	○
Attention to detail in coding and debugging	○	○	○	○	○
Effective use of coding tools and environments	○	○	○	○	○
Knowledge of best practices in code organization and formatting	○	○	○	○	○
Comfort with asking for help and providing feedback	○	○	○	○	○
Persistence in solving challenging problems	○	○	○	○	○
Efficiency in writing clean, maintainable code	○	○	○	○	○

Table 5: Skill improvement questions used in the survey.

Q2 - Awareness of personal strengths and weaknesses in programming: Which skills do you feel have most improved from pair programming and reflections? (Select all that apply)

- Problem-solving skills
- Syntax and code structure understanding
- Debugging and error handling
- Collaboration and communication
- Confidence in coding
- Logical thinking and algorithm development
- Time management and task prioritization
- Familiarity with programming concepts (e.g., loops, functions)
- Adaptability to new coding challenges
- Self-assessment and reflection skills

Q3 - Self-Efficacy in Problem Solving: Rate your confidence in solving programming problems.
Open-Ended: Can you provide an example of a programming problem where pair programming improved your approach?
Q4 - Understanding Core Programming Concepts: Rate your understanding of core programming concepts.
Open-Ended: Which programming concept do you feel most improved through pair programming?
Q5 - Adaptability to New Programming Challenges: Rate your comfort level with tackling unfamiliar programming challenges.
Open-Ended: Share an instance where pair programming helped you overcome an unfamiliar challenge.
Q6 - Ability to Give and Receive Feedback: Rate your confidence in giving constructive feedback.
Open-Ended: Describe one piece of feedback you gave or received that significantly impacted your learning.

Q7 - Engagement and Motivation in Programming: Rate your motivation to engage with programming tasks.
Open-Ended: Which aspect of pair programming did you find most motivating?
Q8 - Reflection on Strengths and Weaknesses: Rate your awareness of your strengths and weaknesses in programming.
Open-Ended: What is the most significant strength or weakness you identified this semester?
Q9 - Perceived Improvement in Programming Efficiency: Rate your efficiency in completing programming tasks.
Open-Ended: What specific strategies improved your efficiency during pair programming?
Q10 - Collaboration Willingness in Programming Projects: Rate your confidence in collaborating with others on programming projects.
Open-Ended: What is one thing you learned about teamwork through pair programming?
Q11 - Enhancement of Learning Experience: Rate your confidence in engaging with structured feedback mechanisms.
Open-Ended: Can you describe a moment where feedback from these sessions helped you learn something better?
Q12 - Self-Expression with the Professor: Rate your confidence in expressing your thoughts and concerns to professors.
Open-Ended: Can you share an example where feedback made you feel seen or understood by your professor?
Q13 - Perception of Being Seen and Understood: Rate how much you felt your feedback was valued.
Open-Ended: How did being able to give feedback impact your overall learning experience?
Q14 - Advocacy Skills in Other Classes: Rate how confident you are in advocating for yourself in other classes.
Open-Ended: Can you provide an example of how feedback practices in this course influenced your approach in another class?
Q15 - Effect of Identifying Learned Skills or Improvements: (Question 1 - Tell me one thing that you learned or improved from today's exercise.) Rate your ability to recognize what you learned or improved.
Open-Ended: Can you share an example of a skill or concept you better understood by reflecting on what you learned during a pair programming session?
Q16 - Effect of Identifying Areas for Practice: (Question 2 - Tell me one thing that you need more practice on.) Rate how aware you were of the areas in programming where you needed more practice.
Open-Ended: Can you provide an example of how identifying an area for practice helped guide your learning?
Q17 - General Impact of Reflection Questions: Rate your ability to engage in self-reflection about your learning and practice.
Open-Ended: How have these reflection questions influenced your approach to programming or problem-solving?

A2. Content Analysis of Weekly Structured Reflections

The results of the feedback from second pair programming section are given in Table 6. This section highlights areas where students are making progress and encountering challenges. Syntax-related issues were a concern for 35.7% of students, with common struggles around semicolons and scanf usage. 28. 6% of the students had a difficult time working with loops and recursive logic.

On the positive side, 23.8% of students appreciated working with partners, as collaboration helped them debug and share ideas effectively. Similarly, 19.0% of students enjoyed the exercises, especially the creative and real-world tasks that kept them engaged. Conceptual understanding stood out as a key area, with 42.9% of students reflecting mixed feelings. They appreciated learning new topics but found advanced concepts like loops and conditionals challenging.

To address these insights, we developed actionable strategies. For syntax and debugging, we introduced focused exercises with immediate feedback to help students correct mistakes on the spot. Collaboration remained a focus,

Table 6: Content Analysis of Second Pair Programming Reflections

Category	Frequency	Example from Responses
Syntax Challenges	15	"I need more practice with syntax and how and when to use certain commands." "Formatting again and learning not to overcomplicate code."
Loops	12	"I need more practice on loops in general. It takes me a while to comprehend or set up the loop." "I need more practice with recursive loops."
Collaboration and Peer Learning	10	"Working with a partner really helps me because some things I might struggle with my partner has a better hold on." "Learning their thought process will benefit me in the future."
Engagement and Fun	8	"The exercise was fun; it was similar to the thermostat problem on homework 2." "I liked how open-ended today's task was."
Conceptual Understanding	18	"I learned how to include if-else statements in while loops and how to use textttscanf and put that into another variable." "I improved my understanding of how while loops break/spit out infinitely."

with structured peer programming to ensure everyone participates equally. Thematic exercises were expanded to maintain student engagement and motivation. For tougher concepts, we provided step-by-step guidance to help students gradually build their confidence.

The third pair programming session, the results of which are presented in Table 7, provided valuable insights into student progress and challenges. Syntax and formatting issues were a recurring theme and highlighted by 23.8% of students, who faced difficulties managing syntax errors and maintaining clean code structure. Loops and logic stood out for 28.6% of students, with many improving their understanding of while loops but expressing a need for more practice with do-while loops.

Collaboration was appreciated by 19.0% of students, who emphasized the benefits of working with partners to debug and develop ideas. Functions and advanced concepts presented challenges for 35.7% of students, as they worked on mastering nested functions and the efficient use of return types. Finally, 21.4% of students enjoyed the real-world relevance of the exercise, particularly coding a game like Blackjack and exploring applications in robotics.

To address these findings, we planned to introduce targeted syntax exercises to help students identify and fix errors more effectively. Additional tutorials on loops, including do-while structures, would provide more opportunities for practice. Collaborative programming remained the focus, with structured peer interactions to encourage teamwork. For advanced concepts, we incorporated step-by-step exercises on nested functions and their practical applications. Finally, we expanded the use of real-world examples to maintain engagement and demonstrate the relevance of programming skills.

The fourth pair programming session, presented in Table 8, highlighted key areas where students made progress and faced challenges. Approximately 26.2% of students mentioned improvement in syntax and formatting, indicating a greater need for targeted practice to reduce common errors. Functions emerged as a primary learning focus, with 31.0% of students expressing both improvements and a desire for additional practice to build confidence. Loops and

Table 7: Content Analysis of Third Pair Programming Reflections

Category	Frequency	Example from Responses
Syntax and Formatting	10	"I learned that formatting is crucial when dealing with multiple edge cases in loops." "My fast typing causes multiple syntax errors."
Loops and Logic	12	"I improved my knowledge on while loops and randomization in programming." "I need more practice using `do-while` loops effectively."
Collaboration and Peer Learning	8	"Working with a partner helped me debug faster and learn new strategies." "We had great discussions about logic that improved the code quality."
Functions and Advanced Concepts	15	"I learned how to create and call functions within other functions." "I need more practice organizing and using functions efficiently in code."
Engagement and Real-Life Application	9	"The exercise was fun and engaging, especially coding Blackjack." "I loved seeing how these skills could apply to robotics and games."

logic were noted by 21.4% of students as an area of growth and challenge, especially with nested and random number-based loops.

Collaboration was mentioned positively by 14.3% of students, with many highlighting the benefits of peer learning and communication. Finally, 24% of students appreciated the connection to real-world applications, finding exercises engaging and relevant to practical coding scenarios.

To address these findings, targeted exercises focused on reducing syntax errors and strengthening students' comfort with functions, loops, and logic. Collaborative practices remained central, with an emphasis on structured teamwork. Real-world inspired exercises continued to motivate and engage students while reinforcing core concepts.

Table 9 presents the data from the fifth pair programming session highlighting the areas of progress and challenges in learning. About 26.2% of students reported improvements in understanding arrays, functions, and pointers, signaling these as critical focus areas. Additionally, 19.0% expressed a need for further practice with formatting and syntax, particularly in using format specifiers and managing curly braces effectively.

Collaboration was appreciated by 14.3% of students, as working with peers helped clarify complex topics. However, 16.7% mentioned that time management during exercises was a concern, with some finding the activities too lengthy. Exercises with real-world relevance were praised by 9.5% of students, emphasizing their motivational impact.

To address these findings, future sessions included focused exercises on arrays, pointers, and functions with immediate feedback. Structured collaboration activities helped students benefit from peer interactions. Time management was an issue for many, with exercises often feeling too lengthy, the time was adjusted, and real-world challenges maintained engagement while ensuring the exercises remained manageable.

The sixth pair programming session given in Table 10 provides areas of progress and challenges in programming exercises. About 26.5% of students reported improved understanding of strings, functions, and pointers, showing these as key areas for focus. Additionally, 14.7% expressed a need for more practice with formatting and syntax,

Table 8: Content Analysis of Fourth Pair Programming Reflections

Category	Frequency	Example from Responses
Syntax and Formatting	11	”I need more practice in functions and syntax.” ”Remembering ’;’ at the end of lines.”
Functions	13	”I learned how to make a function take inputs, how variables are created within a function.” ”I learned how to effectively use functions to make code more readable.”
Loops and Logic	9	”I learned how to use a while loop in a complicated way.” ”I improved on recursive loops today and using the rand function library.”
Collaboration	6	”I improved upon collaboration.” ”I liked today’s exercise because my partner explained things I didn’t understand.”
Real-World Applications	10	”Today’s exercise was fun as it had a ’real’ purpose.” ”I liked today’s exercise especially since it has a real-world use.”

particularly when using format specifiers and debugging errors.

Collaboration was valued by 11.8% of students, as peer programming helped clarify difficult topics and foster teamwork. However, 17.6% mentioned struggles with understanding pointers and their applications, while 8.8% highlighted challenges with time management during exercises. Exercises with real-world relevance were appreciated by 8.8% of students, motivating them to connect coding to practical use cases.

To address these insights, sessions prioritized targeted practice on strings, pointers, and modular functions, offering real-time feedback. Structured peer programming enhanced collaborative learning. Time-managed exercises with real-world applications maintained motivation and ensured tasks were feasible within the allotted time.

The feedback collected from the seventh session presented in Table 11 highlights several areas where students experienced growth, faced challenges, and found value in the exercises. Nearly 30% of students reported challenges with pointers and structs, emphasizing difficulties in passing pointers to functions and effectively using struct pointers in modular code. About 28% of responses focused on file I/O, with students highlighting struggles in using functions like `fgets`, avoiding segmentation faults, and handling file read/write operations.

Syntax and formatting issues were noted by 22% of students, particularly with the arrow operator (−>) and formatting strings while using file operations. Collaboration was appreciated by 15% of students, as peer programming helped them debug and improve their understanding of file operations and structs. Engagement and fun were mentioned by 12% of students, with thematic exercises making the learning process enjoyable and motivating. Debugging and error handling were valued by 18% of students, especially in identifying and resolving issues with file handling and logic errors.

Based on these insights, actionable strategies were developed to address the challenges and reinforce the successful aspects of the learning environment. Targeted exercises were introduced to focus on pointers, structs, and file I/O, with immediate feedback to help students troubleshoot effectively. Debugging tasks were embedded into exercises to simulate real-world scenarios and strengthen problem-solving skills. Structured collaboration activities were maintained to enhance peer learning, and thematic challenges were incorporated to keep students engaged while reinforcing core concepts.

Feedback from the eighth session indicates strong progress in understanding pointers and structs, with 35% of

Table 9: Content Analysis of Fifth Programming Reflections

Category	Frequency	Example from Responses
Arrays, Functions, and Pointers	11	"I learned how to use arrays and pointers in functions." "I improved my knowledge of linking arrays and functions."
Formatting and Syntax	8	"I need more practice formatting tables and using format specifiers." "I improved my use of spacing in `printf` functions."
Collaboration	6	"Working with my partner helped me understand arrays better." "Peer programming made complex tasks easier to manage."
Time Management	7	"The exercise was too lengthy to finish during the lab." "We struggled to manage time effectively with so many functions."
Real-World Relevance	4	"The exercise showed how arrays can be applied in real-world scenarios." "It was fun to work on something that felt practical and meaningful."

students highlighting improvements, as seen in Table 12. File I/O and fgets were challenging, as noted by 28% of students. Syntax and formatting issues, such as handling the arrow operator and newline characters, were reported by 22%. Collaborative exercises were valued by 12% of students, who appreciated peer support. Fun and engaging exercises motivated 10% of participants.

Based on these insights, actionable strategies were developed to address the challenges and reinforce the successful aspects of the learning environment. Targeted exercises were introduced to focus on pointers, structs, and the effective use of fgets, with immediate feedback to help students correct mistakes during the exercises. Structured peer programming activities were maintained to enhance collaboration and engagement. Debugging tasks were incorporated into exercises to address file I/O challenges and teach students error-handling techniques effectively.

The data collected in the ninth pair programming session highlights key areas where students showed growth, faced challenges, and appreciated the learning experience. As presented in 13, around 40% of the responses focused on mastering pointers and structs, particularly nested structs and dereferencing pointers. 35% of responses emphasized challenges and progress with File I/O and the `fgets` function, showcasing the need for further practice in file handling.

Issues with syntax and formatting were noted in 30% of responses, particularly in using operators like the arrow operator (−>). Collaborative activities were praised by 18%, as students found peer programming helpful in understanding difficult concepts. Finally, 15% of responses highlighted engagement and fun, with thematic exercises receiving positive feedback for keeping learning enjoyable and relevant.

To address these findings, future sessions included focused exercises on pointers, structs, and File I/O with immediate feedback to solidify understanding. Structured collaboration activities helped students benefit from peer interactions, enhancing teamwork and problem-solving. Syntax-focused mini-reviews targeted common issues like when to use the dot operator (.) and when to use the arrow operator (−>). Engaging thematic challenges maintained student

Table 10: Content Analysis of Sixth Pair Programming Reflections

Category	Frequency	Example from Responses
Strings	18	”I learned how to efficiently use string functions and `fgets`.” ”Today’s exercise helped me get better at understanding null characters and handling strings.”
Pointers	14	”I learned how to iterate through arrays using pointers.” ”I need more practice understanding how pointers and arrays interact.”
Formatting	10	”I got better at using format specifiers to clean up outputs.” ”Formatting outputs with `%lf` was a useful skill to practice.”
Functions	12	”I improved my ability to call functions and pass arguments effectively.” ”Today’s exercise taught me to separate expressions into modular functions for better code organization.”
Debugging	8	”I learned how to debug pointers and segmentation faults in strings.” ”Finding and fixing errors in code was a valuable part of today’s exercise.”
Collaboration	6	”I enjoyed working with my partner to solve today’s exercise.” ”Peer programming helped me understand different perspectives.”

interest while ensuring exercises were appropriately scoped to fit within the class duration.

The reflections from the last pair programming session, as shown in Table 14, highlighted key areas of growth and challenges. About 40% of the responses indicated significant improvements in understanding pointers and the malloc function, signaling these as crucial focus areas. Around 35% of the responses emphasized challenges related to dynamic memory management, particularly knowing when and how to free memory safely. Syntax and formatting issues were noted in 30% of responses, highlighting the need for practice with precise malloc syntax and output formatting.

Collaboration was valued by 20% of students, as working with peers facilitated better understanding and faster debugging. Lastly, 15% of responses emphasized engagement and fun, particularly in exercises that incorporated thematic or creative challenges to demonstrate malloc’s utility.

To address these insights, future sessions included targeted exercises on dynamic memory allocation, focusing on safe memory handling and real-world applications of malloc. Structured collaboration activities supported peer learning and encouraged teamwork. Syntax-focused mini-reviews tackled common errors in malloc and free usage. Engaging challenges, such as creative use cases for dynamic memory, were designed to sustain student interest and motivation while reinforcing core concepts.

Table 11: Content Analysis of Seventh Pair Programming Reflections

Category	Frequency	Example from Responses
Pointers and Structs	30	”I learned how to use string pointers rather than using character arrays when passing strings into a function.” ”I improved my understanding of passing pointers to functions for better memory efficiency.”
File I/O and fgets	28	”I learned how to avoid segmentation faults when reading/writing .txt files.” ”Understanding fgets helped me handle input more efficiently and avoid common errors.”
Syntax and Formatting	22	”I need more practice understanding the arrow operator and fgets syntax.” ”Formatting structs and correctly using fgets was challenging but insightful.”
Collaboration	15	”Working with partners helped clarify tricky concepts in structs.” ”My partner’s input helped us debug and improve the logic efficiently.”
Engagement and Fun	12	”I enjoyed the animal-based exercise; it was engaging and fun.” ”The thematic exercises made learning complex concepts enjoyable and interactive.”
Debugging and Error Handling	18	”Reading my code more closely helped me identify a typo in a string causing errors.” ”Debugging with printf statements helped me understand logic errors effectively.”
File Operations	20	”I learned how to better open, read, write, and close files in C.” ”Using fscanf and error checking improved my ability to handle file operations correctly.”

Table 12: Content Analysis of Eight Pair Programming Reflections

Category	Frequency	Example from Responses
Pointers and Structs	35	”I learned how to reference structures through their memory addresses.” ”I improved my ability to pass struct pointers into functions effectively.”
File I/O and fgets	28	”I learned how to avoid segmentation faults when working with files.” ”Understanding fgets helped me handle input more efficiently in this exercise.”
Syntax and Formatting	22	”I need more practice understanding the arrow operator and fgets syntax.” ”I often forget to remove newline characters when using fgets.”
Collaboration	12	”Working with partners helped clarify tricky concepts in structs.” ”My partner’s explanations made debugging easier and less frustrating.”
Engagement and Fun	10	”I enjoyed the animal-based exercise; it was engaging and fun.” ”The thematic exercises made the learning process enjoyable and interactive.”

Table 13: Content Analysis of Nineth Pair Programming Reflections

Category	Frequency	Example from Responses
Pointers and Structs	40	”I learned how to reference structures through their memory addresses.” ”I improved my understanding of pointers when working with nested structs.”
File I/O and fgets	35	”I learned how to avoid segmentation faults when working with files.” ”I practiced using `fgets` and understanding its nuances in handling strings.”
Syntax and Formatting	30	”I need more practice understanding the arrow operator and `fgets` syntax.” ”Formatting structs and correctly using `->` where necessary was a challenge for me.”
Collaboration	18	”Working with partners helped clarify tricky concepts in structs.” ”Collaboration made debugging faster and more insightful.”
Engagement and Fun	15	”I enjoyed the animal-based exercise; it was engaging and fun.” ”The thematic exercises made learning concepts like structs interesting.”

Table 14: Content Analysis of Tenth Pair Programming Reflections

Category	Frequency	Example from Responses
Pointers and Malloc	40	”I learned how to better use malloc and memory allocation as well as pointers.” ”I improved my understanding of passing arrays using pointers and malloc.”
Dynamic Memory Management	35	”I feel more comfortable with malloc after today’s exercise.” ”I need more practice knowing when to free memory and avoid leaks.”
Syntax and Formatting	30	”I spent time formatting numbers for the screen to make the output clean and precise.” ”Understanding the proper syntax for malloc and free was challenging but useful.”
Collaboration	20	”Working with my partner helped me solidify malloc concepts.” ”Peer programming made debugging malloc usage errors much faster and easier.”
Engagement and Fun	15	”Today’s exercise was short and fun; it allowed us to explore malloc effectively.” ”I liked the challenge part where we applied malloc in a creative way.”