DOI: https://doi.org/10.31341/jios.44.2.1 UDC 004.42:378-022.215(594)
Open Access Original Scientific Paper

The Impact of Pair Programming on the Performance of Slow-Paced Students: A Study on Data Structure Courses

Mewati Ayub *mewati.ayub@it.maranatha.edu*
Faculty of Information Technology
Maranatha Christian University, Bandung, Indonesia

Oscar Karnalim *oscar.karnalim@it.maranatha.edu*
Faculty of Information Technology
Maranatha Christian University, Bandung, Indonesia

Laurentius Risal *laurentius.risal@it.maranatha.edu*
Faculty of Information Technology
Maranatha Christian University, Bandung, Indonesia

Maresha Caroline Wijanto *maresha.cw@it.maranatha.edu*
Faculty of Information Technology
Maranatha Christian University, Bandung, Indonesia

Abstract

A study shows that pair programming can help slow-paced students in completing Introductory Programming assessment. This paper replicates the study on Data Structure course, in which the completion of the assessments does not only rely on logic but also theoretical knowledge. The aim is to check whether pair programming is still helpful on such new assessment characteristics. Three classes of Data Structure course with 14 teaching weeks and a total of 72 undergraduate students are considered in this study. Two of the classes are about Basic Data Structure while another one is the advanced one. Our evaluation shows that pair programming can help slow-paced students in both pair and individual academic performance. It also increases overall academic performance if the tasks are more logic oriented. Nevertheless, no benefits provided for fast-paced students paired to the slow-paced ones, even though all students appreciate the use of pair programming.

Keywords: pair programming, slow-paced students, data structure, quasi experiment, computer science education

1. Introduction

Learning programming is often perceived as challenging [1],[2],[3]. Many strategies have been developed to deal with such an issue: bootstrapping the materials [4],[5], the use of interactive media [6],[7], and promoting collaboration [8],[9].

Pair programming, which is originally a part of extreme programming [10], is an attempt of promoting collaboration in which two students create a particular solution together [11]. Commonly, one of them acts as the driver who controls all of the shared resources while another acts as the navigator for providing advice. The roles are dynamic in a sense that they can be switched naturally after a period of time [12].

Many studies have explored the impact of pair programming in which the details can be seen in a literature review [13]. Brought, Eby, and Wahls [14] and Nagappan et al. [15] argued that this technique provides higher retention as the students become more confident [16], [14] and enjoy the process [17]. It can also boost up their individual programming skill [14] and the quality of the work [18].

Gomez et al. [19] found that the use of IDE-support tool may affect the effectiveness of pair programming. Another affecting factor is student programming skill. Lewis and Shah [20] showed that high skill difference can lead to faster task completion due to dominance of the more skilled member.

Salleh, Mendes, and Grundy [21] explored that among three factors of personality framework, only openness affects student performance in a statistically significant manner. Villamor and Rodrigo [22] tried to utilise dual eye tracking with Cross-Recurrence Quantification Analysis to predict the success of collaboration on pair programming but failed. The analysis was argued to be insufficient.

Kavitha and Ahmed [23] explored that pair programming can be useful for knowledge sharing at postgraduate level. Saltz and Shamshurin [24] found that pair programming is applicable on data science course, where a software programmer acts as the driver and a data science researcher acts as the navigator.

Yang, Lee, and Chang [25] discovered that in terms of explaining data structure, pair programming might help students in being more confident and having better retention of learning. Chaparro, et al [26] confirmed that in Object-Oriented Programming, students enjoyed being paired as long as their skill gap is not too large. McDowell, Hanks, and Werner [27] encouraged the use of pair programming in completing team-based tasks given that such collaboration enables students to learn from one another.

A meta-analysis was conducted by Hannay et al. [10], comparing pair to solo (or individual) programming. When the task is simple, pair programming can shorten the completion time. Solo programming, on the other hand, can lead to higher code quality on complicated tasks.

Ayub et al. [28] showed that pair programming can be useful to help slow-paced students in learning introductory programming. Per in-class programming assessment, each slow-paced student is paired with a fast-paced one and the latter is encouraged to teach the former. According to the study, it increases the academic performance of both types of students, even though the slow-paced ones experienced more.

This paper replicates Ayub et al.'s study [28] to check whether their technique is also applicable on Data Structure courses, which is argued to have at least three differences to Introductory Programming. First, the programming solution is not only about logic but also theoretical knowledge. Second, there is a possible gap

between algorithmic and programming knowledge given that the lecture is mostly focused on how the data structure works at theoretical level [29]. Third, students enrolled in this course have known programming as this course is typically taken after Introductory Programming [30]. Two classes of Basic Data Structure and a class of Advanced Data Structure were considered in this study, with 14 teaching weeks each and 72 undergraduate students in total.

Our study is different to the one conducted by Yang, Lee, and Chang [25] given that their study is focused on student capability in explaining data structure programs via comments while ours is focused on student capability in implementing data structures and using them to solve a particular case.

2. Methodology

Ayub et al.'s proposed technique [28] has four consecutive stages. Firstly, the students are ranked based on their academic performance in descending order. The first half is considered as fast-paced students while the counterpart is considered as the slow-paced ones. Secondly, each fast-paced student ranked at K^{th} position will be paired to a student at $(N-K+1)^{th}$ position where N refers to the number of students. For example, a 2nd ranked student will be paired with a student who is the second last. This assures each student pair has comparable programming skill. Thirdly, each student pair is asked to collaborate in a programming assessment but with two computers provided. One computer can be primarily used to complete the assessment while another can be used to read the tasks or course materials. While completing the assessment, the slow-paced student is in favour regardless of the role. If they act as an observer, they can see how the fast-paced one solves the task. Otherwise, they can learn by writing the solution as suggested by the fast-paced one. Finally, few slow-paced students will be asked to explain some parts of their solutions in front of the class upon assessment completion. If they are unable to do that, the assessment's score will be reduced, affecting both the student and the fast-paced counterpart. This encourages fast-paced students to teach their corresponding slow-paced student about how the solution works.

The technique was applied on Data Structure courses, in which each weekly programming assessment was about as a large task implementing a data structure and using it on a particular case. This makes the assessments different to the Introductory Programming ones as not only logic is required to create the solution. A sufficient amount of theoretical knowledge about the data structure is also required. The courses cover two levels. Basic level covers simple data structures (e.g., stack, queue, linked list, and priority queue) in which the programming assessments are mostly about translating algorithms to programs with one-to-one conversion (one algorithm instruction to one program statement). Advanced level covers more complex data structures like binary tree and graph. The programming assessments are also about translating algorithms to programs. However, the algorithms are highly abstracted so that the translation process requires higher programming logic than that in basic level.

3. Evaluation

Per Data Structure course, five evaluated factors in Ayub et al.'s study [28] were adapted and reused: pair academic performance of slow-paced students, individual academic performance of slow-paced students, pair academic performance of all students, individual academic performance of all students, and pair academic performance of fast-paced students. The first three are our main focus while the last two are supplementary.

3.1. Evaluation Scenario

Two Basic Data Structure classes (referred as DS1A and DS1B) and one Advanced Data Structure class (DS2) were considered in this study with 72 students in total. The former was conducted on the first semester of 2019 and the latter was conducted on short semester of 2019. Each evaluated factor was addressed by considering two groups of assessments: pair and individual. Pair assessment is Ayub et al.'s proposed technique [28] described in methodology in which the academic performance was measured based on their scores on previous individual assessment. Individual assessment means no collaboration is involved for completing that assessment.

Both groups of assessments were performed alternatively among weekly sessions in each class. The detail of such distribution can be seen on Table 1. Each assessment should be completed in a programming laboratory without internet and removable disk accesses. The maximum completion time per assessment is 100 minutes.

Teaching week	Assessment Type
1	Individual
2	Pair
3	Individual
4	Pair
5	Individual
6	Pair
7	Individual
8	Pair
9	Individual
10	Pair
11	Individual
12	Pair
13	Individual
14	Pair

Table 1. Assessments distribution among weekly sessions

Task given per assessment is mostly about implementing a data structure and applying it in a particular case. For DS1A and DS1B, the weekly topics can be seen in Table 2 and each assessment should be completed with Python programming

language. DS1A has 23 students participated till the end of the course while DS1B has 25 students.

Teaching week	Assessment topic
1	Introduction to Abstract Data Type (ADT)
2	Single ADT
3	Multiple ADT
4	Array ADT
5	Stack with Array Representation
6	Queue with Array Representation
7	Linked List Part 1
8	Linked List Part 2
9	Linked List Part 3
10	Stack and Queue with Linked List Implementation
11	Priority Queue with Linked List Implementation
12	Linked List Variant: Double Pointer
13	Linked List Variant: Circular
14	Shell Sort and Quick Sort

Table 2. Assessment topics of Basic Data Structure

For DS2, the assessment should be completed in Python-like Java with the help of a library created by one of the authors (unpublished). The use of the library is expected to introduce data types and mitigate the burden of transition while the students start learning Java or C# in later courses. This is why some weekly topics displayed in Table 3 are about Introductory Programming review as the students are required to adapt themselves to the library. DS2 has 24 students participating till the end of the course.

Teaching week	Assessment topic
1	Input, Output, and Branching
2	Looping and Function
3	Array and Matrix
4	Recursion
5	ADT Part 1
6	ADT Part 2
7	Binary Search Tree
8	Heap Tree
9	Graph Part 1
10	Graph Part 2
11	Sparse Matrix
12	Hashing
13	ADT in C#
14	ADT in Java

Table 3. Assessment topics of Advanced Data Structure

Three evaluated factors (pair academic performance of slow-paced students, individual academic performance of slow-paced students, and pair academic performance of fast-paced students) are analysed with one of the student groups, either slow-paced or fast-paced. The former is students whose average score is at bottom-half rank list while the remaining students are considered as the latter.

To capture student perspectives about pair programming, a questionnaire survey was also developed and distributed to the students at the end of the semester. Each student was asked to rate eight statements in five-point Likert scale, showing their agreement toward the statements. The scale has five values in which "1" refers to strongly disagree, "3" refers to neutral, and "5" refers to strongly agree. The details of those statements can be seen in Table 4. Among the statements, S5 and S6 are negative statements, which are not expected to be agreed by the students. Unfortunately, we were not able to distribute the survey to Basic Data Structure students, and only Advanced Data Structure students were participated (24 students).

ID	Statement
S1	Pair programming can lead to a constructive discussion in completing the task
S2	Pair programming enables me to help my pair understanding the task and how to complete it
S3	Pair programming enables me to seek help for completing a task
S4	Pair programming encourages me to learn course materials further
S5	Pair programming is time consuming as I need to discuss with my pair in completing the task
S6	Pair programming makes me relying too much to my pair, demotivating me in understanding the task and the solution
S7	Course materials are more understandable if the assessments are completed in pair
S8	Assessment task can be easier to solve when completed in pair

Table 4. Statements used in the questionnaire

The analysis of Basic Data Structure course (DS1A and DS1B) will be discussed first, followed by the analysis of Advanced Data Structure course (DS2). After that, a discussion comparing the findings will be provided. Student perspectives on pair programming will also be discussed based on the questionnaire survey.

3.2. Basic Data Structure: Pair Academic Performance of Slow-Paced Students

This subsection discusses whether pair academic performance of slow-paced students in Basic Data Structure course is improved by the implementation of pair programming. For each pair assessment, slow-paced students' scores were paired to those on previous individual assessment. Fourteen comparisons were collected from two classes of Basic Data Structure course in which the details can be seen in Table 5.

Comparison ID	Class	Pair assessment week	Individual assessment week
CR1101	DS1A	2	1
CR1102	DS1A	4	3
CR1103	DS1A	6	5
CR1104	DS1A	8	7
CR1105	DS1A	10	9
CR1106	DS1A	12	11
CR1107	DS1A	14	13
CR1108	DS1B	2	1
CR1109	DS1B	4	3
CR1110	DS1B	6	5
CR1111	DS1B	8	7
CR1112	DS1B	10	9
CR1113	DS1B	12	11
CR1114	DS1B	14	13

Table 5. Comparisons Made for Measuring Pair Academic Performance Improvement of Slow-Paced Students in Basic Data Structure

Pair programming is effective if the student scores are improved in a statistically significant manner, measured with t-test when the scores are normally distributed (with mean as its determiner) or Wilcoxon Signed Rank for the others. The tests were performed with 95% confidence rate.

As shown in Table 6 with bold P-values representing statistically significant change, only two comparisons (CR1108 to CR1114) shows statistically significant increase, which is 44.18 points improvement in average. Six p-values are not available (NA) due to no score differences between pair and individual sessions; all students completed the tasks perfectly. In other words, pair programming seldom affects pair academic performance of slow-paced students in Basic Data Structure course. However, if it does, the impact is positive.

3.3. Basic Data Structure: Individual Academic Performance of Slow-Paced Students

This section discusses whether individual academic performance of slow-paced students in Basic Data Structure course is improved by the implementation of pair programming. Per pair assessment, slow-paced students' scores on individual assessments before and after the pair assessment were compared one another. Twelve comparisons are considered in which the details can be seen on Table 7.

Pair programming is effective if the student scores are improved in a statistically significant manner upon the completion of a pair assessment. It is measured with t-test when the scores are normally distributed (with mean as its determiner) or Wilcoxon Signed Rank for the others, with 95% confidence rate.

Table 8 shows that three comparisons have no differences, leading the unavailability of the p-values. Another set of three lead statistically significant differences with two of them show an improvement, that is 56 points by average. Pair programming occasionally affects the individual academic performance of

slow-paced students in Basic Data Structure course, and most of such effects are positive.

Comparison ID	Pair Assessment Mean Score	Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR1101	100.00	100.00	-	NA	0.00
CR1102	95.83	95.45	Wilcoxon	0.5	0.38
CR1103	95.83	100	Wilcoxon	1	-4.17
CR1104	100.00	93.64	Wilcoxon	0.125	6.36
CR1105	90.91	100.00	Wilcoxon	0.5	-9.09
CR1106	100.00	100.00	-	NA	0.00
CR1107	100.00	100.00	-	NA	0.00
CR1108	81.00	96.67	Wilcoxon	0.0625	-15.67
CR1109	90.91	92.86	t-test	0.6036	-1.95
CR1110	100.00	50.00	Wilcoxon	**0.0039**	50.00
CR1111	100.00	100.00	-	NA	0.00
CR1112	76.36	38.00	t-test	**<0.0001**	38.36
CR1113	100.00	100.00	-	NA	0.00
CR1114	100.00	100.00	-	NA	0.00

Table 6. Pair Academic Performance Improvement of Slow-Paced Students in Basic Data Structure

Comparison ID	Class	Next Individual Assessment Week	Previous Individual Assessment Week
CR1201	DS1A	3	1
CR1202	DS1A	5	3
CR1203	DS1A	7	5
CR1204	DS1A	9	7
CR1205	DS1A	11	9
CR1206	DS1A	13	11
CR1207	DS1B	3	1
CR1208	DS1B	5	3
CR1209	DS1B	7	5
CR1210	DS1B	9	7
CR1211	DS1B	11	9
CR1212	DS1B	13	11

Table 7. Comparisons Made for Measuring Individual Academic Performance Improvement of Slow-Paced Students in Basic Data Structure

3.4. Basic Data Structure: Pair Academic Performance of All Students

This section discusses whether students' pair academic performance in Basic Data Structure course is improved with pair programming. Every pair assessment was compared to its previous individual assessment in terms of student scores. This mechanism results in 14 comparisons listed on Table 5, but with both slow-paced and fast-paced students on board. Pair programming is effective if pair assessments result in higher score to their corresponding individual assessment in a statistically

significant manner. The significance was measured with t-test for normal distribution and Wilcoxon Signed Rank for others. The tests were carried out with 95% confidence rate.

Comparison ID	Next Individual Assessment Mean Score	Previous Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR1201	95.45	100	Wilcoxon	1	-4.55
CR1202	100	95.45	Wilcoxon	1	4.55
CR1203	93.64	100	Wilcoxon	0.25	-6.36
CR1204	100	93.64	t-test	0.0669	6.36
CR1205	100.00	100	-	NA	0.00
CR1206	100	100.00	-	NA	0.00
CR1207	92.86	96.67	t-test	0.3559	-3.81
CR1208	50	92.86	t-test	0.0558	-42.86
CR1209	100	50	t-test	**0.0043**	50.00
CR1210	38	100	t-test	**<0.0001**	-62.00
CR1211	100	38	t-test	**<0.0001**	62.00
CR1212	100	100	-	NA	0.00

Table 8. Individual Academic Performance Improvement of Slow-Paced Students in Basic Data Structure

Among the comparisons, Table 9 shows that five of them have no p-values due to no difference between the mean scores. For the remaining comparisons, three are significant (see the bold p-values) in which two of them are positive, with 28.92 points improvement. Pair programming can be beneficial for both slow-paced and fast-paced students in Basic Data Structure course, and the impact is commonly positive.

3.5. Basic Data Structure: Individual Academic Performance of All Students

This section discusses whether pair programming increases students' individual academic performance for Basic Data Structure course. Twelve comparisons listed on Table 7 were reused, but with both slow-paced and fast-paced students on board. Pair programming is effective if pair assessments can increase student scores in the next individual assessment in a statistically significant manner, measured with t-test for normal distribution and Wilcoxon Signed Rank for others. Both are with 95% confidence rate.

Table 10 shows that four comparisons (see the bold ones) are statistically significant with half of them leads to improvement (36.3 points by average). No strong findings can be gained as the numbers of positive and negative comparisons are the same.

Comparison ID	Pair Assessment Mean Score	Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR1101	100	100	-	NA	0.00
CR1102	98	97.92	Wilcoxon	0.5	0.08
CR1103	98	100	Wilcoxon	1	-2.00
CR1104	100	97.08	Wilcoxon	0.125	2.92
CR1105	95.83	100	Wilcoxon	0.5	-4.17
CR1106	100	100	-	NA	0.00
CR1107	100	100	-	NA	0.00
CR1108	88.64	97.14	Wilcoxon	**0.0078**	-8.51
CR1109	95.65	97.37	Wilcoxon	0.25	-1.72
CR1110	100.00	67.39	Wilcoxon	**0.0002**	32.61
CR1111	100.00	100.00	-	NA	0.00
CR1112	85.22	60.00	t-test	**0.0001**	25.22
CR1113	100.00	100.00	-	NA	0.00
CR1114	100.00	100.00	-	NA	0.00

Table 9. Pair Academic Performance Improvement of All Students in Basic Data Structure

Comparison ID	Next Individual Assessment Mean Score	Previous Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR1201	97.92	100.00	Wilcoxon	1	-2.08
CR1202	100	97.92	Wilcoxon	1	2.08
CR1203	97.08	100	Wilcoxon	0.25	-2.92
CR1204	100	97.08	Wilcoxon	0.125	2.92
CR1205	100	100	-	NA	0.00
CR1206	100	100	-	NA	0.00
CR1207	97.37	97.14	Wilcoxon	0.5	0.23
CR1208	67.39	97.37	Wilcoxon	**0.0039**	-29.98
CR1209	100.00	67.39	Wilcoxon	**0.002**	32.61
CR1210	60.00	100.00	Wilcoxon	**0.0002**	-40.00
CR1211	100.00	60.00	t-test	**<0.0001**	40.00
CR1212	100.00	100.00	-	NA	0.00

Table 10. Individual Academic Performance Improvement of All Students in Basic Data Structure

3.6. Basic Data Structure: Pair Academic Performance of Fast-Paced Students

This section discusses whether pair academic performance of fast-paced students is also improved with pair programming on board. Per pair assessment, the scores of fast-paced students were compared to those from the previous individual assessment. Fourteen comparisons displayed in Table 5 were reused but with only fast-paced

students on board. Pair programming is effective if the score improvement is statistically significant, measured with t-test for normal distribution and Wilcoxon Signed Rank for unnormal distribution, with 95% confidence level.

As shown in Table 11, eleven of fourteen comparisons provide no differences. Among the remaining three, only one is statistically significant and it is positive (CR1112 which p-value is bold). In other words, no benefits gained by fast-paced students for Basic Data Structure course, probably due to the fact that they can solve the tasks by themselves.

Comparison ID	Pair Assessment Mean Score	Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR1101	100.00	100.00	-	NA	0.00
CR1102	100.00	100.00	-	NA	0.00
CR1103	100.00	100.00	-	NA	0.00
CR1104	100.00	100.00	-	NA	0.00
CR1105	100.00	100.00	-	NA	0.00
CR1106	100.00	100.00	-	NA	0.00
CR1107	100.00	100.00	-	NA	0.00
CR1108	95.00	97.5	Wilcoxon	0.25	-2.50
CR1109	100.00	100.00	-	NA	0.00
CR1110	100.00	83.33	Wilcoxon	0.125	16.67
CR1111	100.00	100.00	-	NA	0.00
CR1112	93.33	78.33	t-test	**0.0102**	15.00
CR1113	100.00	100.00	-	NA	0.00
CR1114	100.00	100.00	-	NA	0.00

Table 11. Individual Academic Performance Improvement of Fast-Paced Students in Basic Data Structure

3.7. Basic Data Structure: Finding Summary

For Basic Data Structure course, pair programming can help slow-paced students in terms of both pair and individual academic performance. Pair assessments often have higher marks than the counterpart. However, no strong relations are found between pair programming and individual academic performance of all students. It also shows no benefits for fast-paced students as they can get perfect scores in both pair and individual assessments.

3.8. Advanced Data Structure: Pair Academic Performance of Slow-Paced Students

The impact of pair programming on pair academic performance of slow-paced students in Advanced Data Structure course is discussed in this section. Similar to Section 3.2, each pair assessment was compared to the previous individual assessment in terms of slow-paced student scores. Further, the significance was measured with either t-test (for normal distribution) or Wilcoxon Signed Rank (for

others). Both are with 95% confidence rate. Seven comparisons were made for this purpose in which the details can be seen in Table 12.

Comparison ID	Pair Assessment Week	Individual Assessment Week
CR2101	2	1
CR2102	4	3
CR2103	6	5
CR2104	8	7
CR2105	10	9
CR2106	12	11
CR2107	14	13

Table 12. Comparisons Made for Measuring Pair Academic Performance Improvement of Slow-Paced Students in Advanced Data Structure

Four of seven comparisons are statistically significant (see the bold p-values on Table 13), with three of them are positive, improving the scores by 11.05 points in average. For this course, slow-paced students occasionally experience score improvement with pair programming on board.

Comparison ID	Pair Assessment Mean Score	Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR2101	97.92	100.00	t-test	**0.0172**	-2.08
CR2102	92.56	78.18	t-test	**0.0006**	14.38
CR2103	96.67	82.22	t-test	**0.0436**	14.44
CR2104	85.00	78.33	t-test	0.2291	6.67
CR2105	92.08	84.17	t-test	0.1843	7.92
CR2106	97.52	93.18	Wilcoxon	**0.0039**	4.34
CR2107	97.73	100.00	t-test	0.0531	-2.27

Table 13. Pair Academic Performance Improvement of Slow-Paced Students in Advanced Data Structure

3.9. Advanced Data Structure: Individual Academic Performance of Slow-Paced Students

The impact of pair programming on individual academic performance of slow-paced students in Advanced Data Structure course is discussed in this section. Following Section 3.3, for each pair assessment, individual assessments prior and upon the pair assessment were compared in terms of slow-paced student scores. Only statistically significant differences are discussed in which the significance was determined with t-test or Wilcoxon Signed Rank, with 95% confidence rate. The former was applied for normal distribution while the latter was for the others. Table 14 shows six comparisons made for this purpose.

Three comparisons yield statistically significant differences (see bold p-values on Table 15). Two of them are positive with 7.89 points improvement. Hence, it can

be stated that pair programming increases individual academic performance of slow-paced students in some cases.

Comparison ID	Next Individual Assessment Week	Previous Individual Assessment Week
CR2201	3	1
CR2202	5	3
CR2203	7	5
CR2204	9	7
CR2205	11	9
CR2206	13	11

Table 14. Comparisons Made for Measuring Individual Academic Performance Improvement of Slow-Paced Students in Advanced Data Structure

Comparison ID	Next Individual Assessment Mean Score	Previous Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR2201	78.18	100.00	t-test	**0.0001**	-21.82
CR2202	82.22	77.22	t-test	0.3748	5.00
CR2203	80.00	84.00	t-test	0.5017	-4.00
CR2204	84.17	78.33	t-test	0.1801	5.83
CR2205	93.18	82.73	t-test	**0.0231**	10.45
CR2206	100.00	94.50	t-test	**0.0318**	5.50

Table 15. Individual Academic Performance Improvement of Slow-Paced Students in Advanced Data Structure

3.10. Advanced Data Structure: Pair Academic Performance of All Students

The impact of pair programming on pair academic performance in Advanced Data Structure course is discussed in this section in a similar way as Section 3.4 except that the course is Advanced Data Structure. Per pair assessment, it was compared to its previous individual assessment in terms of student scores, with significant differences discussed. Significance was measured under 95% confidence rate with

Comparison ID	Pair Assessment Mean Score	Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR2101	98.33	100.00	Wilcoxon	**0.0078**	-1.67
CR2102	95.26	86.74	Wilcoxon	**0.0001**	8.52
CR2103	96.82	91.82	Wilcoxon	**0.0039**	5.00
CR2104	85.83	84.38	T-test	0.652	1.46
CR2105	92.92	92.08	Wilcoxon	0.6438	0.83
CR2106	97.73	96.30	Wilcoxon	**<0.0001**	1.42
CR2107	98.33	100.00	Wilcoxon	0.0781	-1.67

Table 16. Pair Academic Performance Improvement of All Students in Advanced Data Structure

t-test (for normal distribution) or Wilcoxon Signed Rank. Seven comparisons listed on Table 12 were reused for this evaluation.

Table 16 shows that more than half comparisons are statistically significant (see the bold p-values) in which three quarters of those are positive (with 4.98 points average improvement). Hence, pair programming has an impact on students' pair academic performance in Advanced Data Structure course and the impact is often positive.

3.11. Advanced Data Structure: Individual Academic Performance of All Students

The impact of pair programming on individual academic performance in Advanced Data Structure course is discussed in this section. The analysis was carried similarly as Section 3.5 except that the course is Advanced Data Structure. For each pair assessment, its adjacent individual assessments were compared, expecting the one upon the pair assessment gains higher score. T-test was used to measure the significance on normal distribution with 95% confidence rate. Otherwise, Wilcoxon Signed Rank was used with the same confidence rate. All comparisons listed on Table 14 were reused for this.

Table 17 shows that all comparisons are statistically significant (all p-values are bold) with two thirds of them are positive (5.18 points improvement). This means that on Advanced Data Structure course, pair programming is likely to increase individual academic performance of all students.

Comparison ID	Next Individual Assessment Mean Score	Previous Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR2201	86.74	100.00	Wilcoxon	**0.0002**	-13.26
CR2202	91.82	86.74	Wilcoxon	**0.0181**	5.08
CR2203	84.38	91.82	Wilcoxon	**0.0315**	-7.44
CR2204	92.08	84.38	Wilcoxon	**0.0052**	7.71
CR2205	96.30	92.08	Wilcoxon	**0.0117**	4.22
CR2206	100.00	96.30	Wilcoxon	**0.0156**	3.70

Table 17. Individual Academic Performance Improvement of All Students in Advanced Data Structure

3.12. Advanced Data Structure: Pair Academic Performance of Fast-Paced Students

The impact of pair programming on pair academic performance of fast-paced students in Advanced Data Structure course is discussed in this section. This is similar to Section 3.6 except that the course is Advanced Data Structure. Each pair assessment was compared to its previous individual assessment based on fast-paced students' scores. Significance was measured using t-test with 95% confidence rate

for normal distribution, and Wilcoxon Signed Rank with the same confidence rate for others. Seven comparisons listed on Table 12 were reused for this evaluation.

As seen in Table 18 with bold p-values depicting significant changes, three comparisons are statistically significant but only one of those is positive. Hence, it can be stated that pair programming tends to reduce pair academic performance of fast-paced students.

Comparison ID	Pair Assessment Mean Score	Individual Assessment Mean Score	Significance Test	P-Value	Improvement
CR2101	98.75	100.00	Wilcoxon	0.25	-1.25
CR2102	97.73	94.58	Wilcoxon	**0.0039**	3.14
CR2103	97.08	98.33	Wilcoxon	0.125	-1.25
CR2104	86.67	90.42	t-test	0.2668	-3.75
CR2105	93.75	100.00	t-test	**0.0242**	-6.25
CR2106	98.48	99.17	Wilcoxon	**0.002**	-0.68
CR2107	98.75	100.00	Wilcoxon	0.5	-1.25

Table 18. Pair Academic Performance Improvement of Fast-Paced Students in Advanced Data Structure

3.13. Advanced Data Structure: Finding Summary

For Advanced Data Structure course, pair programming can increase academic performance of students (regardless pair or individual). However, fast-paced students may not benefit from it and have their pair academic performance reduced.

3.14. Generalised Findings from Basic and Advanced Data Structure courses

From both courses, it is clear that pair programming is helpful for slow-paced students, in both pair and individual academic performance. However, fast-paced students may get no benefits, or even lower performance if the tasks require complex logic (like the ones in Advanced Data Structure course). Fast-paced students might take too much effort in explaining the task and the solution as simple as possible to their corresponding slow-paced student.

Pair programming can also increase academic performance of all students in general, but it is exclusive to Advanced Data Structure that requires more logic. Hence, we believe such benefit is proportional to how complex the logic needed for solving the tasks. This is expected as many slow-paced students in programming lack of strong logical thinking, and they can seek help from the fast-paced ones during pair programming.

Compared to Ayub et al.'s study [28], the findings are somewhat similar except that on Data Structure courses, pair programming has stronger impact in increasing individual academic performance.

3.15. Student Perspectives on Pair Programming

Figure 1 shows that all positive statements were agreed by the respondents. S3 ("pair programming enables me to seek help for completing a task") gains the highest average score and it is in accordance to a generalised finding from the experiment, stating that pair programming can help slow-paced students in completing the task. The lowest-scored statement is S7 ("course materials are more understandable if the assessments are completed in pair") even though it still tends to agree as the average score is higher than '3' (neutral). It is possible that slow-paced students relied too much to the fast-paced ones or the discussion took too much time. These are slightly in accordance to the results of S5 ("pair programming is time consuming as I need to discuss with my pair in completing the task") and S6 ("pair programming makes me relying too much to my pair, demotivating me in understanding the task and the solution"), negative statements which are weakly disagreed by the students.

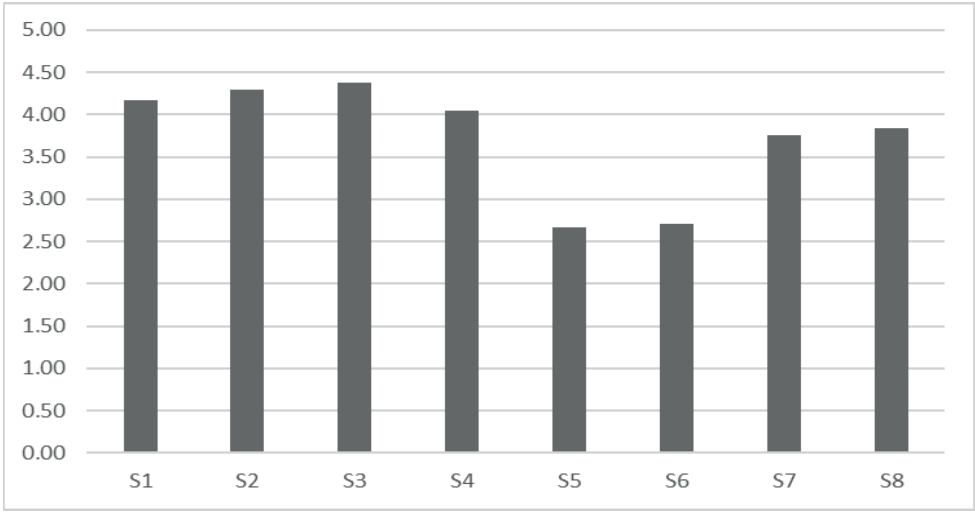

Figure 1. Questionnaire Results

4. Conclusions

This paper replicates Ayub et al.'s study [28] on Data Structure courses in which the assessments should be completed not only with logic but also theoretical knowledge. Three classes of Data Structure courses (two basic and one advance) are considered in which each class has 14 teaching weeks and 72 undergraduate students in total.

Our evaluation shows that pair programming can help slow-paced students in terms of both pair and individual academic performance. However, no advantages are provided for the fast-paced ones, and sometimes this can lead to reduced pair academic performance. This is in accordance with Ayub et al.'s study [28]. Pair programming can also be beneficial for increasing academic performance of all students in general if the tasks require more complex logic.

Our survey shows that students responded positively to pair programming as it accommodates slow-paced students in seeking help for understanding and completing a task. However, some of them believe this technique can be time consuming and can lead to over-reliance from slow-paced students to their corresponding fast-paced pair.

As our future work, we plan to investigate empirical factors supporting the success of pair programming. This can be beneficial for those who are interested in applying the technique to help slow-paced students in programming, as the chance of successful implementation can be improved.

Acknowledgements

The author(s) received a financial support for the publication of this article from Maranatha Christian University, Indonesia.

References

[1] C. McDonald, *Why Is Teaching Programming Difficult?* in Higher Education Computer Science. Springer International Publishing, 2018.

[2] M. Piteira, and C. Costa, " Learning computer programming: Study of difficulties in learning programming, " *International Conference on Information Systems and Design of Communication*, 2013, pp. 75–80.

[3] Y. Qian, and J. Lehman, "Students' misconceptions and other difficulties in introductory programming: A literature review," *ACM Transactions on Computing Education*, vol. 18, no.1, pp. 1-24, 2017.

[4] G. Kiss, and Z. Arki, "The Influence of Game-based Programming Education on the Algorithmic Thinking," *Procedia - Social and Behavioral Sciences*, vol. 237, pp. 613–617, 2017.

[5] P. Mullins, D. Whitfield, and M. Conlon, "Using Alice 2.0 as a first language," *Journal of Computing Sciences in Colleges*, vol. 24, no. 3, pp. 136–143, 2009.

[6] O. Debdi, M. Paredes-Velasco, and J.A. Velázquez-Iturbide, " GreedExCol, A CSCL tool for experimenting with greedy algorithms," *Computer Applications in Engineering Education*, vol. 23, no. 5, pp. 790–804, 2015.

[7] Elvina, O. Karnalim, M. Ayub, and M.C. Wijanto, "Combining program visualization with programming workspace to assist students for completing programming laboratory task," *Journal of Technology and Science Education*, vol. 8, no. 4, pp. 268–280, 2018.

[8] P. Lasserre, "Adaptation of team-based learning on a first term programming class," *Proceeding of the 14th Annual ACM SIGCSE*

Conference on Innovation and Technology in Computer Science Education, 2009, pp. 186-190.

[9] P. Lasserre, and C. Szostak, "Effects of team-based learning on a CS1 course," *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education - ITiCSE '11*, 2011, pp.133-137.

[10] J. E., Hannay, T. Dybå, E. Arisholm, dan D.I.K. Sjøberg, "The effectiveness of pair programming: A meta-analysis," *Information and Software Technology*, vol. 51, no. 7, pp. 1110–1122, 2009.

[11] K. Beck, and E. Gamma, *Extreme programming eXplained : embrace change*. Addison-Wesley, 2000.

[12] L. Williams, and R.R. Kessler, *Pair programming illuminated*. Addison-Wesley, 2003.

[13] N. Salleh, E. Mendes, and J. Grundy, "Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 37, no. 4, pp. 509–525, 2011.

[14] G. Braught, L.M. Eby, and T. Wahls, "The effects of pair-programming on individual programming skill," *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '08*, 2008, pp. 200-204.

[15] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik, "Improving the CS1 experience with pair programming," *ACM SIGCSE Bulletin*, vol. 35, no. 1, pp. 359–362, 2003.

[16] S. B. Berenson, K. M. Slaten, L. Williams, and C.-W. Ho, "Voices of women in a software engineering course: reflections on collaboration," *Journal on Educational Resources in Computing*, vol. 4, no. 1, pp. 3-21, 2004.

[17] E. Mendes, L.B. Al-Fakhri, and A. Luxton-Reilly, "Investigating pair-programming in a 2nd -year software development and design computer science course," *ACM SIGCSE Bulletin*, vol. 37, no. 3, pp. 296–300, 2005.

[18] L.A. Williams, dan R.R. Kessler, "The effects of "pair-pressure" and "pair-learning" on software engineering education," *Thirteenth Conference on Software Engineering Education and Training*, 2000, pp. 59–65.

[19] O. S. Gomez, A.A. Aguileta, R.A. Aguilar, J.P. Ucan, R.H. Rosero, dan K. Cortes-Verdin, "An empirical study on the impact of an IDE tool support in the pair and solo programming," *IEEE Access*, vol. 5, pp. 9175–9187, 2017.

[20] C. M. Lewis, and N. Shah, "How equity and inequity can emerge in pair programming," *Proceedings of the Eleventh Annual International Conference on International Computing Education Research - ICER '15*, 2015, pp. 41–50.

[21] N. Salleh, E. Mendes, and J. Grundy, "Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments," *Empirical Software Engineering*, vol. 19, no. 3, pp. 714–752, 2014.

[22] M. Villamor, and M. M. Rodrigo, "Predicting successful collaboration in a pair programming eye tracking experiment," *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization - UMAP '18*, 2018, pp. 263–268.

[23] R. K. Kavitha, and M. S. I. Ahmed, "Knowledge sharing through pair programming in learning environments: An empirical study," *Education and Information Technologies*, vol. 20, no. 2, pp. 319–333, 2015.

[24] J. S. Saltz, and I. Shamshurin, "Does pair programming work in a data science context? An initial case study," *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 2348–2354.

[25] Y. F. Yang, C.I. Lee, and C. K. Chang, "Learning motivation and retention effects of pair programming in data structures courses," *Education for Information*, vol. 32, no. 3, pp. 249–267, 2016.

[26] E. A. Chaparro, A. Yuksel, P. Romero, and S. Bryant, "Factors affecting the perceived effectiveness of pair programming in higher education," *17th Workshop of the Psychology of Programming Interest Group*, Sussex University, 2005, pp. 5–18.

[27] C. E. McDowell, B. F. Hanks, and L. Werner, "Experimenting with pair programming in the classroom," *ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education*, 2003, pp. 60–64.

[28] M. Ayub, O. Karnalim, Risal, W. F. Senjaya, and M. C. Wijanto, "Utilising pair programming to enhance the performance of slow-paced students on Introductory Programming," *Journal of Technology and Science Education*, vol. 9, no. 3, pp. 357–367, 2019.

[29] R. A. Nathasya, O. Karnalim, and M. Ayub, "Integrating program and algorithm visualisation for learning data structure implementation," *Egyptian Informatics Journal*, vol. 20, no. 3, pp. 193–204, 2019.

[30] A. M. Abirami, and P. Kiruthiga, "Collaborative learning tools for data structures," *Journal of Engineering Education Transformations*, vol. 31, no. 3, pp.79–83, 2018.