Hindawi
Education Research International
Volume 2021, Article ID 9105342, 10 pages
https://doi.org/10.1155/2021/9105342

Research Article

A Study of First-Year Students' Attitudes toward Programming in the Innovation in Educational Technology Course

Virawan Amnouychokanant,[1] **Surapon Boonlue**,[2] **Saranya Chuathong**,[2] **and Kuntida Thamwipat**[2]

[1]*Division of Learning Innovation and Technology, Faculty of Industrial Education and Technology, King Mongkut's University of Technology Thonburi, 126 Pracha Uthit Road, Bang Mod, Thung Khru, Bangkok 10140, Thailand*
[2]*Department of Educational Communications and Technology, Faculty of Industrial Education and Technology, King Mongkut's University of Technology Thonburi, 126 Pracha Uthit Road, Bang Mod, Thung Khru, Bangkok 10140, Thailand*

Correspondence should be addressed to Virawan Amnouychokanant; virawan.am@mail.kmutt.ac.th

Received 24 June 2021; Accepted 29 September 2021; Published 14 October 2021

Academic Editor: Enrique Palou

The purpose of this study was to investigate the relationship between students' attitudes toward programming, gender, and learning performances. The survey used for measuring students' attitudes toward programming consisted of 20 questions on a five-point Likert scale in five dimensions (meaningfulness, interest in programming, self-efficacy, creativity, and collaboration). Ninety freshmen who had basic programming experience by using block-based programming in the Innovation in Educational Technology course were asked to take the survey. The overall reliability of the survey was found to be 0.93. The results showed that there was no significant difference between male and female freshmen in attitude toward programming, but there was a significant difference among different learning performances in dimensions of interest in programming, self-efficacy, and creativity. We performed pairwise comparisons at the same level of significance by using Fisher's least significant difference (LSD) method to test which group differs from the other groups. The results found that low-performing students' attitudes toward programming in dimensions of interest in programming, self-efficacy, and creativity were the lowest of all types of students. This is a challenge for instructors in planning learning activities to encourage low-performing students to have a more positive attitude toward programming.

1. Introduction

Programming is an essential skill for students in the digital era. The world economy is being transformed, and more and more workers are being replaced by robots and artificial intelligence (AI). Consequently, programming is one of the high-demand skills in current and future job markets [1]. Many people think programming skills are only valuable for people working in highly technical careers. In fact, for people who work closely with developers and programmers, learning basic programming makes them much more valuable members of a team. Within programming in the higher education context, students learn how to break down a problem into smaller parts and design a step-by-step procedure for creating a working program by using a language that the computer understands [2]. These processes related to decomposition and algorithm design in computational thinking give students new perspectives on problem-solving. Furthermore, learning to code offers students the opportunity to have great earning potential in the future and makes them become in-demand candidates in a rapidly shifting digital economy [3, 4]. Nowadays, Thailand is concentrating on digital technology as a solid foundation for future business and as the main driving force in educational reform. Thailand has developed a national socioeconomic and educational development plan for human resources with an eye to the job market of the future [5]. To support this plan, higher education institutions must produce graduates

who are highly skilled in digital areas such as programming, artificial intelligence, machine learning, data science, data analytics, cloud computing, blockchain, cybersecurity, mobile development, user interface design, and user experience design [6–12].

One member of our research team is an instructor who teaches Innovation in Educational Technology. He wanted to promote programming among students in his course because he realized that programming is one of the high-demand skills in the present and future. As a result of completing the Innovation in Educational Technology course, students will be able to apply basic programming and computational thinking concepts to create innovative inventions. This course is not a specialization in computer science. Therefore, students do not need to be as proficient in programming as computer science students, but they need to have basic knowledge of programming to apply it in the future. In this course, block-based programming is considered as an alternative to foster basic programming skills because text-based coding is not easy for beginners to start coding and the language syntax is a barrier for students' understanding of computational thinking concepts [13]. Furthermore, block-based languages have a pallet of commands, making memorizing commands unnecessary; therefore, it is easy for novices [14]. To measure students' attitudes toward programming, students were asked to take the survey after completing the course.

Many studies explored the attitudes toward programming of male and female students [15–18]. On the other hand, the attitudes toward programming of students who have different levels of performance have been scarcely studied. The purpose of this study is to provide empirical evidence that can help to answer a set of research questions: what are the attitudes toward programming of male and female students? What are the attitudes toward programming of students who have different levels of performance (high, medium, and low)? This paper is organized as follows: Section 2 reviews the background literature on programming in higher education and components of attitudes toward programming; Section 3 outlines the methodology, including the participants, instrument, and procedure; Section 4 presents the results of the survey; Section 5 summarizes the discussion; and Section 6 presents the conclusion.

2. Literature Review

2.1. Programming in Higher Education. Introductory programming is important for undergraduate students in the twenty-first century. Previous skills and background knowledge are key for a novice student to learn basic programming. The skills were divided into two categories: programming-related skills and general educational skills [19]. The goal of introductory programming courses is to teach the students how to express solutions in a language that the computer understands. Choosing an appropriate language for teaching basic programming is critical. It has a strong effect on novice students. If they understand the first language they learn, they tend to continue coding and learn more difficult programming languages [20]. Many researchers emphasize that learning how to program requires the use of problem-solving skills [21–23]. A lack of problem-solving skills makes it difficult to learn programming [24]. Therefore, it is considered a prerequisite skill for learning basic programming. In addition to problem-solving skills, mathematical ability is cited as necessary for programming. Most programming teachers state that lacking mathematical ability is one of the difficulties that novice students encounter when programming. When students do not have enough mathematical skills, they do not know how to program [25]. It is noteworthy that students who have mathematical skills and logical reasoning always succeed in a programming course [26]. Porter and Zingaro [27] consider previous knowledge as a predictor of success in programming for novice learners in higher education. Students who had prior programming experience tend to perform better than those with no programming experience. Concerning general educational skills, lacking knowledge of English is an obstacle in programming because it is used in the syntax of all programming languages. Students who are good at vocabulary and English grammar were more likely to be successful programmers than nonfluent ones [28].

Learning to program in higher education is difficult because it requires problem-solving and higher-order thinking skills [29–31]. The difficulty of expressing the solution in a language understood by the computer is a barrier for novice students. They need to understand many abstract terms. Even students who have enough problem-solving skills find it hard to turn the pseudocode into a syntactically correct program. The syntax of programming languages is important for novice students. Correcting syntax errors is a time-consuming process and leads to random debugging behavior [32]. Selecting the appropriate control structures (sequences, loops, conditionals, recursion, and repetition) for solving problems was also mentioned as one of the difficulties for novice students in higher education [32–42]. Bringula et al. [43] found that most students failed the programming examination because the level of difficulty of the examination is not suitable to the students' level and allotted time. When they often cannot solve programming errors, they experience anxiety, panic, and stress [44]. They also sorrow and despair [45]. Finally, they tend to give up and transfer to other degree programs when they cannot overcome programming difficulties [31].

Basic programming is a challenge not only for novice students but also instructors. Instructors attempt to find ways to arouse students' interest in programming [46]. The style of teaching that does not attract students' interest can make them bored. Making higher education students interested in programming is not an easy task. This is a challenge for instructors when designing learning activities. In addition to motivating the students, instructors should provide novices with fundamental skills and simple problem solving before starting the course so that novices have the background needed to understand basic programming. Previous studies have suggested various teaching methods: live coding, gamification, team based-learning, interactive computer tutors, mentor support, peer instruction, and pair

programming [47–54]. However, some of these teaching methods do not always succeed. Hertz and Jump [55] used a variety of methods in the classroom, including active learning, demonstration, live coding, and canned examples. Yet student performance was poor and retention rates were low. In contrast, using only a trace-driven teaching method decreased the dropout rate and grade failures. It can be interpreted as showing that combining various teaching approaches in courses may not always lead to positive results. Moreover, it is difficult for the instructors to foster the development of programming skills in large classes with students who have different levels of knowledge and learning styles [46]. Some educators [56–58] suggested that tutors and mentors would help make learning activities more fluid and fewer differences in individual students. Another challenge is the feedback process. Formative and summative assessment can give quality feedback. In the formative assessment, the instructors can get feedback about their teaching and use the feedback to improve it. The instructors also use summative assessment to determine whether the students are ready to move onto the next lesson and help identify weak areas for students [59]. Lastly, choosing the first programming language to be taught to novice students is significant. The instructors should consider easy languages for beginners to start coding because the language syntax is a barrier for them. Choosing languages that are too difficult can affect students' performance and their attitudes toward programming [13, 14, 20].

2.2. Components of Attitude toward Programming. An understanding of students' attitudes toward programming may help to develop better teaching tools and methods. Students' attitudes toward programming also affect their performance in programming. In this study, we focus on five components of attitudes toward programming including (1) meaningfulness, (2) interest in programming, (3) self-efficacy, (4) creativity, and (5) collaboration.

(1) Meaningfulness is a student's perceived value of programming [60]. When students perceive programming as meaningful, they will put more effort to overcome obstacles when programming [61]. (2) Interest in programming is the state of wanting to learn about programming. Students with an interest in programming tend to have better performance than other students. They are willing to spend more time on programming. They are more likely than others to view difficult programming tasks as a challenge and find effective solutions to complete it [62]. (3) Self-efficacy is students' belief in their ability to succeed in programming [63]. When students have greater self-efficacy, they have greater confidence in their ability to overcome programming difficulties and are more likely to continue working on it until completion [61]. (4) Creativity is a student's perception that he or she could try different methods and ideas when faced with problems in programming [61]. Although programming is related to logical reasoning, students also believe creativity is important in programming [64]. Computational thinking and programming skills can develop through creative programming. Creative programming encourages the students in the process of designing and developing work through coding [65]. (5) Collaboration is a student feeling toward doing programming activities with peers. Programming has become a collaborative task because today's programs are too difficult and long for a single programmer to complete. Therefore, students should be familiar with collaboration before making careers in the future [66]. Pair programming plays an important role in higher education. Working in pairs produces more rapid and effective solutions than working alone. In addition, students with high programming skills will help a partner who has lower programming skills to complete their task [67]. Getting students to write code together in pairs or small groups can also enhance students' programming performance and confidence [52–54]. Students with greater collaboration attitudes tend to make more effort and more efficiently solve problems when working collaboratively with others [61].

3. Methodology

3.1. Participants. The respondents of this study were 90 freshmen attending the Department of Educational Communications and Technology of King Mongkut's University of Technology Thonburi (KMUTT), Thailand. All of them enrolled in the Innovation in Educational Technology course. Before taking an online survey, all respondents had basic programming experience by using block-based programming [68]. In this study, we classified the students into three groups: high-, medium-, and low-performing students. The students' classification details are shown in Table 1.

3.2. Instrument and Procedure. This survey drew on the questionnaire developed by Kong et al. [61]. Some items of their questionnaire were not directly related to freshmen's attitude toward programming. Since our purpose was to develop the instrument to measure freshmen's attitude toward programming, we omitted some items in their survey and changed some to reflect our purpose. Finally, we constructed 25 items for our survey. Items were designed using a 5-point Likert scale (5 = "strongly agree," 4 = "agree," 3 = "neutral," 2 = "disagree," and 1 = "strongly disagree"). Three experts in computer science and educational assessment checked the items to ensure content validity. After editing the questionnaire according to the recommendations of the experts, nine freshmen (four males and five females) read the questionnaire to check if they understand the intended meanings of each item. Finally, 90 freshmen in the Innovation in Educational Technology course were asked to take the online survey.

Exploratory factor analysis was used to reduce data to a smaller set of summary variables. Five items were deleted due to insignificant factor loading. There were 20 items left in five factors. Factors were named as "Meaningfulness," "Interest in programming," "Self-efficacy," "Creativity," and "Collaboration." Table 2 shows Cronbach's alpha value of the survey. The reliability for the total scale was 0.93 and the reliability of each dimension ranged from 0.80 and 0.89.

Table 1: Students' classification.

GPa	Level of performers
3.01 to 4.00	High
2.01 to 3.00	Medium
Less than 2.01	Low

Table 2: Cronbach's α value of each dimension.

Dimensions	Number of items	Reliability
Meaningfulness	4	0.82
Interest in programming	4	0.89
Self-efficacy	4	0.84
Creativity	4	0.80
Collaboration	4	0.83
Total	20	0.93

Cronbach's alpha values of 0.7 or higher indicate acceptable internal consistency [69]. Thus, each dimension of this survey indicated acceptable reliabilities of the questionnaire's scales. The survey items translated into English are shown in Table 3.

4. Results

4.1. Attitudes toward Programming Results by Gender. Table 4 shows an independent samples *t*-test comparing gender differences in attitudes toward programming. There was no significant difference between male and female freshmen in attitudes toward programming. The average scores in attitude toward programming of males and females were very close. In Table 5, there was no significant difference between male and female freshmen in all dimensions. However, the average scores of the meaningfulness dimension were the highest. It can be assumed that both male and female freshmen realized the importance of programming. While the average scores of the self-efficacy dimension were the lowest. It indicates that both male and female freshmen lacked confidence in their ability to program.

Figure 1 shows box plots for total scores of attitudes in each dimension split by genders. Total attitude scores in each dimension equal 20 points. Both males and females have the highest scores in meaningfulness. The interquartile range (Q3–Q1) of the meaningfulness in males is shorter than in females. Hence, it can be stated that the attitude scores in the meaningfulness of males have less variability than among females. In addition to meaningfulness, attitude scores in the creativity of males also have less variability than among females. However, the attitude scores in interest in programming and self-efficacy of males have more variability than females. Furthermore, collaboration is the shortest interquartile range among all dimensions, especially in females. It means that the attitude of males and females toward collaboration in programming has the least variability of all five dimensions. However, outliers are marked most in collaboration. It indicates that there are abnormally high and low attitude scores in collaboration.

Table 3: Survey items of the instrument.

Items
1. Meaningfulness
1.1. Programming is useful to me.
1.2. Programming makes me solve problems systematically.
1.3. Programming helps me improve my thinking skills.
1.4. Programming offers the opportunity to have great earning potential.
2. Interest in programming
2.1. Programming is catching my attention.
2.2. The content of programming is fun.
2.3. I am enthusiastic when it is programming time.
2.4. I am very passionate about programming activities.
3. Self-efficacy
3.1. I can learn how to code rapidly.
3.2. I have confidence in my ability to code.
3.3. I can find and resolve programming errors.
3.4. I am good at programming.
4. Creativity
4.1. Programming is a creative activity.
4.2. Programming can inspire me to try new ideas to complete tasks.
4.3. It is necessary to use creativity in programming.
4.4. An appropriate programming environment can develop my creativity.
5. Collaboration
5.1. I like to code with my friends.
5.2. I have better ideas when I code with friends.
5.3. I can help my friends when they face problems in programming.
5.4. I finish tasks faster when I code with my friends.

Table 4: Independent samples *t*-test comparing gender differences in attitude toward programming.

Gender	*N*	Mdn	*M*	SD	*t*	*p*
Male	31	4	3.86	0.80	0.08	0.94
Female	59	4	3.85	0.69		

Table 5: Independent samples *t*-test comparing gender differences in dimensions of attitude toward programming.

Dimensions	Gender	*N*	Mdn	*M*	SD	*t*	*p*
Meaningfulness	Male	31	4	3.86	0.80	0.49	0.62
	Female	59	4	3.85	0.69		
Interest in programming	Male	31	4	3.81	0.86	0.07	0.94
	Female	59	4	3.82	0.75		
Self-efficacy	Male	31	3	3.50	0.73	1.00	0.32
	Female	59	4	3.62	0.61		
Creativity	Male	31	4	3.74	0.75	0.53	0.59
	Female	59	4	3.81	0.62		
Collaboration	Male	31	4	4.10	0.67	1.49	0.14
	Female	59	4	3.93	0.59		

4.2. Attitude toward Programming Results per Learning Performances. Table 6 shows the medians (more representative than the mean in asymmetric distribution), means, and standard deviation of attitudes toward

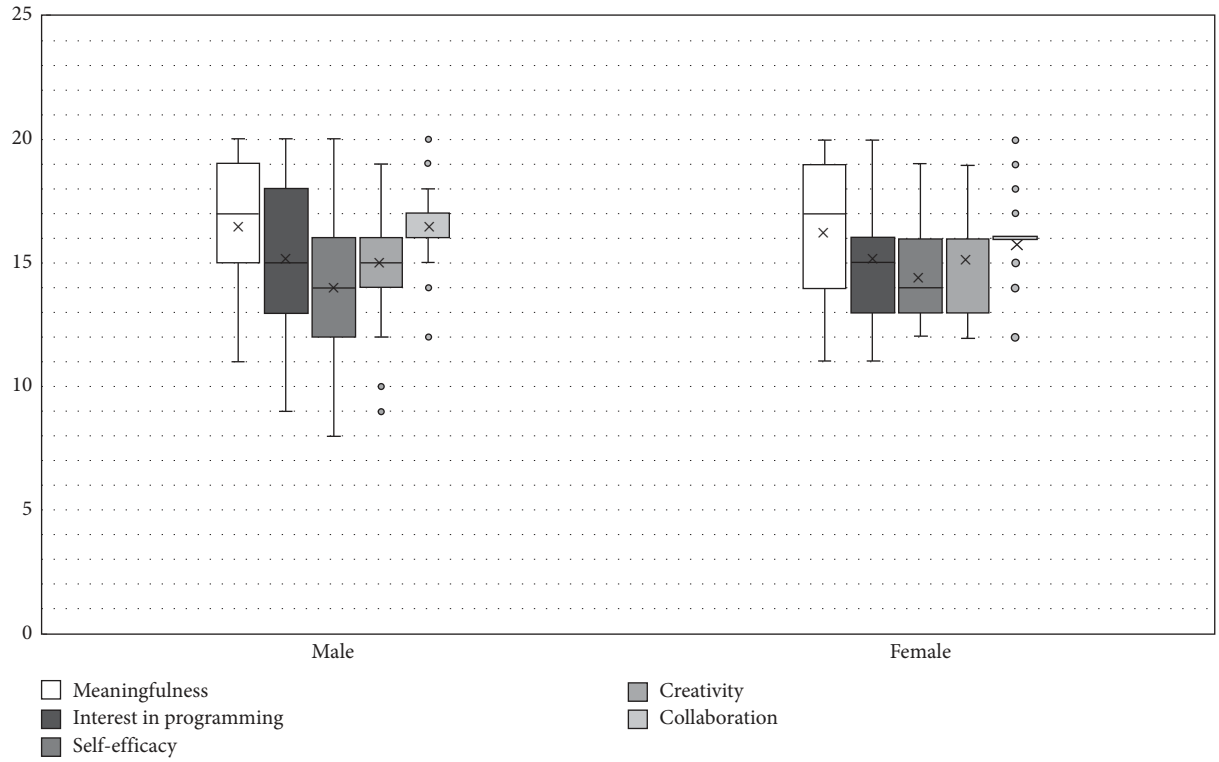

Figure 1: Box plots for total scores of attitudes in each dimension split by gender.

Table 6: Independent samples *t*-test comparing gender differences in dimensions of attitude toward programming.

Learning performances	*N*	Mdn	*M*	SD
High	45	4	3.96	0.67
Medium	43	4	3.78	0.73
Low	2	3	2.90	0.93

Table 7: Analysis of variance (ANOVA) results for attitude toward programming among different learning performances.

Source of the variance	SS	df	MS	*F*	*p*
Between groups	2.567	2	1.283		
Within groups	15.875	87	0.183	7.033	0.001*
Total	18.442	89	—		

Notes: * $p < 0.05$.

programming among students with different learning performances (high, medium, and low). The results found that high-performing students had the most positive attitudes toward programming of all types of students, while low-performing students had the lowest average scores in attitudes toward programming. Table 7 shows a significant difference among students with different learning performances in attitudes toward programming. In Table 8, there was a significant difference among students with different learning performances in dimensions of interest in programming, self-efficacy, and creativity. We performed pairwise comparisons at the same level of significance by using Fisher's least significant difference (LSD) method to test which group differs from the other groups. In Table 9, the results found that both high- and medium-performing students had more interest in programming than low-performing students. Concerning self-efficacy in programming, Table 10 shows a significant difference in all pairs. High-performing students had more confidence in programming than medium- and low-performing students, while medium-performing students had more confidence in programming than low-performing students. In Table 11, the results found that both high- and medium-performing students had a more positive attitude in creativity in programming than low-performing students.

Figure 2 shows box plots for total scores of attitudes in each dimension of attitude toward programming split by level of learning performance. The interquartile range of all dimensions in low-performing students (except for the interquartile range of collaboration) is lower than for high- and medium-performing students. The interquartile range of collaboration in low-performing students is at the same level as for high- and medium-performing students. These results can be interpreted as showing that low-performing students have the same positive attitude in collaboration in programming as high- and medium-performing students. Attitudes toward collaboration in low-performing students have the highest scores among all dimensions. We might assume that low-performing students like to code with peers because it relieves anxiety when facing problems in programming and allows them to finish tasks faster. Outliers are found in high- and medium-performing students but not in low-performing students.

Table 8: Analysis of variance (ANOVA) results among different learning performances in each dimension.

Dimensions	Source of the variance	SS	df	MS	F	p
Meaningfulness	Between groups	1.924	2	0.962	2.541	0.085
	Within groups	32.926	87	0.378		
	Total	34.85	89	—		
Interest in programming	Between groups	4.348	2	2.174	5.150	0.008*
	Within groups	36.722	87	0.422		
	Total	41.07	89	—		
Self-efficacy	Between groups	8.482	2	4.241	21.422	≤0.001*
	Within groups	17.224	87	0.198		
	Total	25.706	89	—		
Creativity	Between groups	4.902	2	2.451	10.596	≤0.001*
	Within groups	20.123	87	0.231		
	Total	25.025	89	—		
Collaboration	Between groups	0.894	2	0.447	1.821	0.168
	Within groups	21.345	87	0.245		
	Total	22.239	89	—		

Notes: * $p < 0.05$.

Table 9: Pairwise comparisons of interest in programming among different learning performances.

Learning performances	M	High 3.97	Medium 3.71	Low 2.63
High	3.97	—	0.26	1.34*
Medium	3.71	−0.26	—	1.08*
Low	2.63	−1.34*	−1.08*	—

Notes: * $p < 0.05$.

Table 10: Pairwise comparisons of self-efficacy in programming among different learning performances.

Learning performances	M	High 3.85	Medium 3.35	Low 2.38
High	3.85	—	0.50*	1.47*
Medium	3.35	−0.50*	—	0.97*
Low	2.38	−1.47*	−0.97*	—

Notes: * $p < 0.05$.

Table 11: Pairwise comparisons of creativity in programming among different learning performances.

Learning performances	M	High 3.91	Medium 3.72	Low 2.38
High	3.91	—	0.19	1.53*
Medium	3.72	−0.19	—	1.34*
Low	2.38	−1.53*	−1.34*	—

Notes: * $p < 0.05$.

5. Discussion

The purpose of this study is to provide empirical evidence that can help to answer a set of research questions: what are the attitudes toward programming for male and female students? What are the attitudes toward programming of students who have levels of different performance (high, medium, and low)? The result showed that there were no significant differences between male and female freshmen in attitudes toward programming. This corresponds to the result of Karaci [15] that indicated that there was no significant difference between male and female students of computer engineering in attitudes toward programming. However, this result is inconsistent with the result of Baser [16] that showed that females' average scores of programming attitude were significantly lower than those of males. In this study, we found the average scores of the meaningfulness dimension were the highest. It can be assumed that both male and female freshmen realized the importance of programming, while the average scores of the self-efficacy dimension were the lowest. That indicates that both male and female freshmen lacked confidence in their ability to program. Therefore, this is a challenge for instructors in planning learning activities to encourage students to have more confidence in programming. When students have greater self-efficacy, they have greater confidence in their ability to overcome obstacles when programming and are more likely to continue working on it until completion [61]. Some educators realize the importance of students' programming self-efficacy. Tsai [70] used block-based programming as an intervention. The results indicated that block-based programming can increase students' programming self-efficacy, especially in students with moderate and low self-efficacy.

In this study, there was a significant difference among different learning performances in dimensions of interest in programming, self-efficacy, and creativity. The result also showed that high-performing students had the most positive attitudes toward programming of all types of students, while low-performing students had the lowest average scores in attitudes toward programming. In terms of creativity, both high- and medium-performing students had a more positive attitude toward creativity in programming than low-performing students. It can be assumed that high- and medium-performing students believed that they could create new ideas while programming. This result corresponds to the result of Kong et al. [61] that indicated that students who have greater creative self-efficacy are more likely to try different methods and ideas when faced with problems in programming. In relation to an interest in programming,

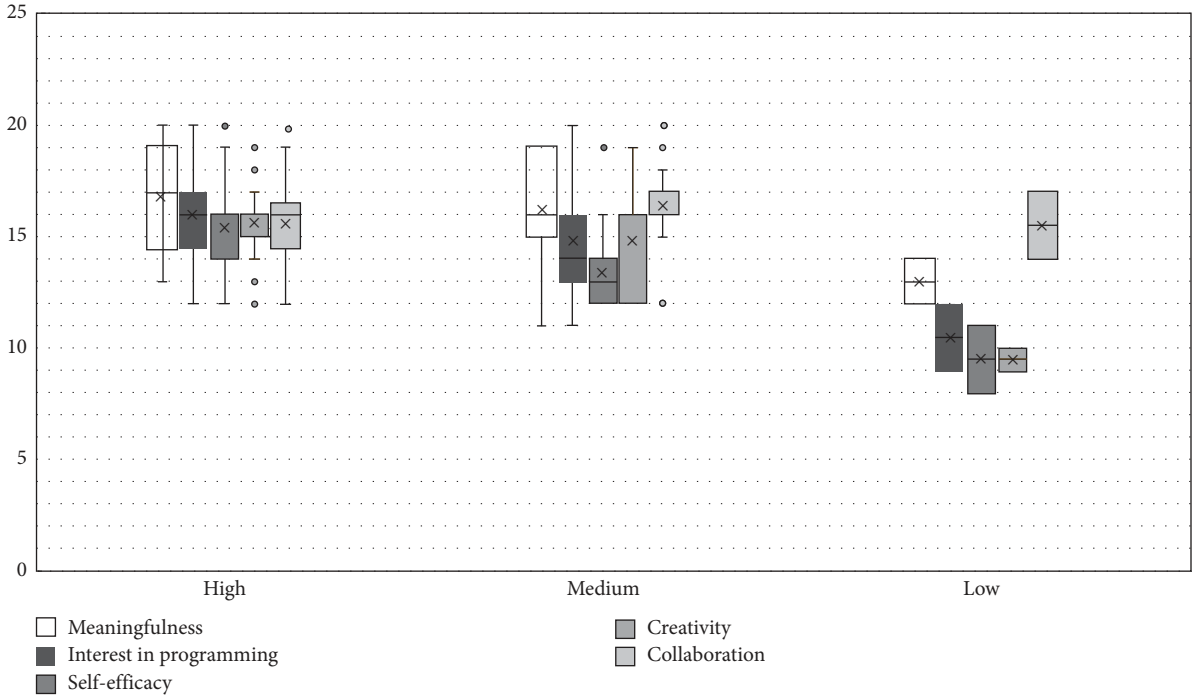

Figure 2: Box plots for total scores of attitudes in each dimension split by learning performances.

both high- and medium-performing students had more interest in programming than low-performing students. Students' interest is crucial. Learning programming will be difficult and boring if they are not interested in the subject. The way the instructors conduct the class also can be a factor that makes students uninterested in the subject [71]. Students who have different performances have different levels of interest and motivation. Thus, the instructors need to consider ways of teaching programming to attract the attention of students, especially low-performing students who are more likely to lack interest in programming than other types of students. Many educators have tried to find ways to increase students' interest and basic knowledge in programming. For instance, Jawad [72] suggested that Android development could increase students' interest in programming. Xu and Jin [73] proposed that game development workshops delivered by peer mentors could increase student curiosity and interest in an introductory programming course. Pradhan [74] suggested that an open-source electronics platform based on easy-to-use hardware and software such as Arduino could increase performance and interest in programming for first-year engineering students. Concerning collaboration, low-performing students in this study had a positive attitude toward collaboration. Therefore, allowing students to practice writing code together in pairs or small groups can improve students' programming performance and confidence [52–54]. Working in pairs produces more rapid and effective solutions than working alone. In addition, students with high programming abilities will assist their partners with low programming abilities in completing their work [67].

Students' attitudes affect performance in programming. When students have a more positive attitude toward programming, they have higher performance in programming [16]. Therefore, the instructors need to know the students' attitude to design teaching methods, materials, and learning activities appropriately.

6. Conclusion

This study used quantitative data to examine students' attitudes toward programming in the Innovation in Educational Technology course. The results showed that there was no significant difference between male and female freshmen in attitudes toward programming, but there was a significant difference among students with different learning performances. Low-performing students' attitudes toward programming in dimensions of interest in programming, self-efficacy, and creativity were the lowest of all types of students. Although there were few low-performing students in the class, they should not be neglected. The curriculum-makers and instructors should find appropriate ways to improve students' attitudes toward programming. When they are interested and confident in programming, they will also achieve a good performance in programming.

6.1. Implications. This study indicates challenges for curriculum-makers and instructors in planning and designing courses to encourage positive student attitudes toward programming. The findings show that low-performing students had the lowest average scores in attitudes toward programming and had self-efficacy, creativity, and interest in programming significantly less than high- and medium-performing students. However, low-performing students in this study had a positive attitude toward collaboration.

Therefore, getting students to write code together in pairs or small groups can enhance students' programming performance and confidence [52–54]. Pair programming produces more rapid and effective solutions than solo programming [75]. Furthermore, students with high programming skills will help a partner who has lower programming skills to complete their task [67]. Curriculum-makers and instructors should pay attention to the student attitudes toward programming because negative student attitudes toward programming can increase the dropout rate and grade failures. Students who lack interest and confidence in programming were more likely to fail the programming examination and eventually give up studying programming [76]. In a nutshell, curriculum-makers and instructors need to be aware of the importance of the student attitudes toward programming and provide proper learning materials and strategies to encourage positive student attitudes toward programming.

6.2. Limitations and Future Studies. The limitation of this study is its sample. This sample only includes freshmen from the Department of Educational Communications and Technology of King Mongkut's University of Technology Thonburi. It is not a representative sample of freshmen in Thailand. Future studies can examine freshmen from more universities in Thailand. In addition, further research may use qualitative data collection and analysis to get more insights into the attitudes toward programming of students. A face-to-face interview would help with noticing the body language of students. Open-ended questions allow respondents to give the data with more diversity than is possible with a closed-question or forced-choice survey measure.

Data Availability

The data used to support this study are included within the article and are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

The authors would like to express sincere thanks to Petchra Pra Jom Klao Ph.D. Research Scholarship from King Mongkut's University of Technology Thonburi for supporting the expense of research.

References

[1] S. W. Kim and Y. Lee, "The effect of robot programming education on attitudes towards robots," *Indian Journal of Science and Technology*, vol. 9, no. 24, pp. 1–11, 2016.

[2] L. M. Pang, H. Ishibuchi, and K. Shang, "Decomposition-based multi-objective evolutionary algorithm design under two algorithm frameworks," *IEEE Access*, vol. 8, pp. 163197–163208, 2020.

[3] K. Dengler and B. Matthes, "The impacts of digital transformation on the labour market: substitution potentials of occupations in Germany," *Technological Forecasting and Social Change*, vol. 137, pp. 304–316, 2018.

[4] S. Fayer, A. Lacey, and A. Watson, "STEM occupations: past, present, and future," *Spotlight on Statistics*, vol. 1, pp. 1–35, 2017.

[5] The Office of the Education Council, "National scheme of education," 2020, https://www.onec.go.th/us.php/home/category/CAT0001145.

[6] A. Jaiswal, C. J. Arun, and A. Varma, "Rebooting employees: upskilling for artificial intelligence in multinational corporations," *International Journal of Human Resource Management*, pp. 1–30, 2021.

[7] D. Vrontis, M. Christofi, V. Pereira, S. Tarba, A. Makrides, and E. Trichina, "Artificial intelligence, robotics, advanced technologies and human resource management: a systematic review," *International Journal of Human Resource Management*, pp. 1–30, 2021.

[8] M.-H. Huang, R. Rust, and V. Maksimovic, "The feeling economy: managing in the next generation of artificial intelligence (AI)," *California Management Review*, vol. 61, no. 4, pp. 43–65, 2019.

[9] G. Petropoulos, "The impact of artificial intelligence on employment," *Praise for Work in the Digital Age*, p. 119, 2018.

[10] J. Djumalieva and C. Sleeman, "Which digital skills do you really need?," 2018, http://coneixement.ctecno.cat/sites/default/files/publicos/Which_digital_skills_do_you_really_need.pdf.

[11] E. van Laar, A. J. A. M. van Deursen, J. A. G. M. van Dijk, and J. de Haan, "21st-century digital skills instrument aimed at working professionals: conceptual development and empirical validation," *Telematics and Informatics*, vol. 35, no. 8, pp. 2184–2200, 2018.

[12] G. Peng, "Do computer skills affect worker employment? an empirical study from CPS surveys," *Computers in Human Behavior*, vol. 74, pp. 26–34, 2017.

[13] D. Topalli and N. E. Cagiltay, "Improving programming skills in engineering education through problem-based game projects with scratch," *Computers & Education*, vol. 120, pp. 64–74, 2018.

[14] D. Weintrop and U. Wilensky, "Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms," *Computers & Education*, vol. 142, Article ID 103646, 2019.

[15] A. Karaci, "Investigation of attitudes towards computer programming in terms of various variables," *International Journal of Programming Languages and Applications*, vol. 6, no. 1/2, pp. 1–9, 2016.

[16] M. Baser, "Attitude, gender and achievement in computer programming," *Online Submission*, vol. 14, no. 2, pp. 248–255, 2013.

[17] M. Derya Gurer, I. Cetin, and E. Top, "Factors affecting students' attitudes toward computer programming," *Informatics in Education*, vol. 18, no. 2, pp. 281–296, 2019.

[18] M. S. Gunbatar and H. Karalar, "Gender differences in middle school students' attitudes and self-efficacy perceptions towards mblock programming," *European Journal of Educational Research*, vol. 7, no. 4, pp. 925–933, 2018.

[19] R. P. Medeiros, G. L. Ramalho, and T. P. Falcão, "A systematic literature review on teaching and learning introductory programming in higher education," *IEEE Transactions on Education*, vol. 62, no. 2, pp. 77–90, 2018.

[20] M. Ateeq, H. Habib, A. Umer, and M. U. Rehman, "C++ or python? which one to begin with: a learner's perspective," in *Proceedings of the 2014 International Conference on Teaching and Learning in Computing and Engineering)*, pp. 64–69, Kuching, Malaysia, April 2014.

[21] R. Mathew, S. I. Malik, and R. M. Tawafak, "Teaching problem solving skills using an educational game in a computer programming course," *Informatics in Education*, vol. 18, no. 2, pp. 359–373, 2019.

[22] S. I. Malik, R. Mathew, and M. M. Hammood, "PROBSOL: a web-based application to develop problem-solving skills in introductory programming," in *Smart Technologies And Innovation For a Sustainable Future*, pp. 295–302, Springer, Cham, Germany, 2019.

[23] A. K. Veerasamy, D. D'Souza, R. Lindén, and M.-J. Laakso, "Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses," *Journal of Computer Assisted Learning*, vol. 35, no. 2, pp. 246–255, 2019.

[24] O. D. L. Tavares, C. S. de Menezes, and R. A. de Nevado, "Pedagogical architectures to support the process of teaching and learning of computer programming," in *Proceedings of the 2012 Frontiers in Education Conference Proceedings*, pp. 1–6, Seattle, WA, USA, October 2012.

[25] A. J. Gomes and A. J. Mendes, "A study on student performance in first year CS courses," in *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, pp. 113–117, Ankara, Turkey, June 2010.

[26] A. Gomes and A. J. Mendes, "Studies and proposals about initial programming learning," in *Proceedings of the 2010 IEEE Frontiers in Education Conference (FIE)*, pp. 1–6, Washington, DC, USA, October 2010.

[27] L. Porter and D. Zingaro, "Importance of early performance in CS1: two conflicting assessment stories," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pp. 295–300, Atlanta, GA, USA, March 2014.

[28] D. Horton and M. Craig, "Drop, fail, pass, continue: persistence in CS1 and beyond in traditional and inverted delivery," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pp. 235–240, Kansas City, MO, USA, February 2015.

[29] M. Mladenović, M. Rosić, and S. Mladenović, "Comparing elementary students' programming success based on programming environment," *International Journal of Modern Education and Computer Science*, vol. 8, no. 8, pp. 1–10, 2016.

[30] M. O. Pendergast, "Teaching introductory programming to IS students: java problems and pitfalls," *Journal of Information Technology Education: Research*, vol. 5, pp. 491–515, 2006.

[31] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: a review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, 2003.

[32] T. Koulouri, S. Lauria, and R. D. Macredie, "Teaching introductory programming: a quantitative evaluation of different approaches," *ACM Transactions on Computing Education*, vol. 14, no. 4, p. 26, 2014.

[33] N. Eltegani and L. Butgereit, "Attributes of students engagement in fundamental programming learning," in *Proceedings of the 2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, pp. 101–106, Khartoum, Sudan, September 2015.

[34] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang, "Ability-training-oriented automated assessment in introductory programming course," *Computers & Education*, vol. 56, no. 1, pp. 220–226, 2011.

[35] S. Xinogalos, "Designing and deploying programming courses: strategies, tools, difficulties and pedagogy," *Education and Information Technologies*, vol. 21, no. 3, pp. 559–588, 2016.

[36] T. Sirkiä and J. Sorva, "Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises," in *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, pp. 19–28, Koli, Finland, November 2012.

[37] L. Ma, J. Ferguson, M. Roper, and M. Wood, "Investigating and improving the models of programming concepts held by novice programmers," *Computer Science Education*, vol. 21, no. 1, pp. 57–80, 2011.

[38] H. Aris, "Improving students performance in introductory programming subject: a case study," in *Proceedings of the 2015 10th International Conference on Computer Science & Education (ICCSE)*, pp. 657–662, Cambridge, UK, July 2015.

[39] M. Yamamoto, T. Sekiya, and K. Yamaguchi, "Relationship between programming concepts underlying programming skills," in *Proceedings of the 2011 International Conference on Information Technology Based Higher Education and Training*, pp. 1–7, Izmir, Turkey, August 2011.

[40] R. Cacceffo, S. Wolfman, K. S. Booth, and R. Azevedo, "Developing a computer science concept inventory for introductory programming," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pp. 364–369, Memphis, TN, USA, February 2016.

[41] A. Robins, "Learning edge momentum: a new account of outcomes in CS1," *Computer Science Education*, vol. 20, no. 1, pp. 37–71, 2010.

[42] A. Berglund and R. Lister, "Introductory programming and the didactic triangle," in *Proceedings of the Twelfth Australasian Conference on Computing Education*, pp. 35–44, Brisbane, Australia, January 2010.

[43] R. P. Bringula, A. D. V. Aviles, M. Y. C. Batalla, and M. T. F. Borebor, "Factors affecting failing the programming skill examination of computing students," *International Journal of Modern Education and Computer Science*, vol. 9, no. 5, pp. 1–8, 2017.

[44] C. Rogerson and E. Scott, "The fear factor: how it affects students learning to program in a tertiary environment," *Journal of Information Technology Education: Research*, vol. 9, pp. 147–171, 2010.

[45] S. Shuhidan, M. Hamilton, and D. D'souza, "A taxonomic study of novice programming summative assessment," in *Proceedings of the Eleventh Australasian Conference on Computing Education*, pp. 147–156, Wellington, New Zealand, January 2009.

[46] A. Gomes and A. Mendes, "A teacher's view about introductory programming teaching and learning: difficulties, strategies and motivations," in *Proceedings of the 2014 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8, Madrid, Spain, October 2014.

[47] R. Ventura Roque-Hernández, S. Armando Guerra-Moya, and F. Carmina Caballero-Rico, "Acceptance and assessment in student pair-programming: a case study," *International Journal of Emerging Technologies in Learning*, vol. 16, no. 9, pp. 4–19, 2021.

[48] R. Shen, D. Wohn, and M. Lee, "Programming learners' perceptions of interactive computer tutors and human teachers," *International Journal of Emerging Technologies in Learning*, vol. 15, no. 9, pp. 123–142, 2020.

[49] N. Zacharis, "Evaluating the effects of virtual pair programming on students' achievement and satisfaction," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 4, no. 3, pp. 34–39, 2009.

[50] S. I. Malik, M. Al-Emran, R. Mathew, R. M. Tawafak, and G. AlFarsi, "Comparison of E-learning, M-learning and game-based learning in programming education—a gendered analysis," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, no. 15, pp. 133–146, 2020.

[51] B. T. Hieu and S. D. S. Mustapha, "Automated data-driven hint generation in intelligent tutoring systems for code-writing: on the road of future research," *International Journal of Emerging Technologies in Learning*, vol. 13, no. 9, pp. 174–189, 2018.

[52] Z. Nurbekova, V. Grinshkun, G. Aimicheva, B. Nurbekov, and K. Tuenbaeva, "Project-based learning approach for teaching mobile application development using visualization technology," *International Journal of Emerging Technologies in Learning*, vol. 15, no. 8, pp. 130–143, 2020.

[53] Z. Nurbekova, T. Tolganbaiuly, P. Tazabekova, G. Abildinova, and B. Nurbekov, "Enhance students' motivation to learn programming through projects," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, no. 21, pp. 133–144, 2020.

[54] Z. Nurbekova, T. Tolganbaiuly, B. Nurbekov, A. Sagimbayeva, and Z. Kazhiakparova, "Project-based learning technology: an example in programming microcontrollers," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, no. 11, pp. 218–227, 2020.

[55] M. Hertz and M. Jump, "Trace-based teaching in early programming courses," in *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, pp. 561–566, Denver, CO, USA, March 2013.

[56] U. Nikula, O. Gotel, and J. Kasurinen, "A motivation guided holistic rehabilitation of the first programming course," *ACM Transactions on Computing Education*, vol. 11, no. 4, p. 24, 2011.

[57] M. Hertz and S. M. Ford, "Investigating factors of student learning in introductory courses," in *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, pp. 195–200, Denver, CO, USA, March 2013.

[58] M. Apiola, N. Moisseinen, and M. Tedre, "Results from an action research approach for designing CS1 learning environments in Tanzania," in *Proceedings of the 2012 Frontiers in Education Conference (FIE)*, pp. 1–6, Seattle, WA, USA, October 2012.

[59] B. Hartanto, "Enhancing the student engagement in an introductory programming: a holistic approach in improving the student grade in the informatics department of the university of Surabaya," in *Communications in Computer and Information Science*, R. Intan, C. H. Chi, H. Palit, and L. Santoso, Eds., vol. 516, pp. 493–504, Springer, Berlin, Germany, 2015.

[60] M. H. Hur, "Empowering the elderly population through ICT-based activities," *Information Technology & People*, vol. 29, no. 2, pp. 318–333, 2016.

[61] S.-C. Kong, M. M. Chiu, and M. Lai, "A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education," *Computers & Education*, vol. 127, pp. 178–189, 2018.

[62] R. M. Ryan and E. L. Deci, "Intrinsic and extrinsic motivation from a self-determination theory perspective: definitions, theory, practices, and future directions," *Contemporary Educational Psychology*, vol. 61, Article ID 101860, 2020.

[63] M. H. Hur, "Empowerment in terms of theoretical perspectives: exploring a typology of the process and components across disciplines," *Journal of Community Psychology*, vol. 34, no. 5, pp. 523–540, 2006.

[64] A. P. Ambrósio, F. M. Costa, L. Almeida, A. Franco, and J. Macedo, "Identifying cognitive abilities to improve CS1 outcome," in *Proceedings of the 2011 Frontiers in Education Conference (FIE)*, pp. F3G-1–F3G-7, Rapid City, SD, USA, October 2011.

[65] M. Romero, A. Lepage, and B. Lille, "Computational thinking development through creative programming in higher education," *International Journal of Educational Technology in Higher Education*, vol. 14, no. 1, pp. 1–15, 2017.

[66] Ö. Demir and S. S. Seferoglu, "The effect of determining pair programming groups according to various individual difference variables on group compatibility, flow, and coding performance," *Journal of Educational Computing Research*, vol. 59, no. 1, pp. 41–70, 2021.

[67] J. T. Nosek, "The case for collaborative programming," *Communications of the ACM*, vol. 41, no. 3, pp. 105–108, 1998.

[68] V. Amnouychokanant, S. Boonlue, S. Chuathong, and K. Thamwipat, "Online learning using block-based programming to foster computational thinking abilities during the COVID-19 pandemic," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 16, no. 13, pp. 227–247, 2021.

[69] J. C. Nunnally, *Psychometric Theory 3E*, Tata McGraw-Hill Education, New York, NY, USA, 1994.

[70] C.-Y. Tsai, "Improving students' understanding of basic programming concepts through visual programming language: the role of self-efficacy," *Computers in Human Behavior*, vol. 95, pp. 224–232, 2019.

[71] M. Rahmat, S. Shahrani, R. Latih, N. F. M. Yatim, N. F. A. Zainal, and R. A. Rahman, "Major problems in basic programming that influence student performance," *Procedia—Social and Behavioral Sciences*, vol. 59, pp. 287–296, 2012.

[72] H. M. Jawad, "Android mobile app development as a motivation towards computer programming," in *Proceedings of the 2019 IEEE International Conference on Electro Information Technology (EIT)*, pp. 169–175, Brookings, SD, USA, May 2019.

[73] X. Xu and W. Jin, "Game development workshops designed and delivered by peer mentors to increase student curiosity and interest in an introductory programming course," in *Proceedings of the 2021 ACM Southeast Conference*, pp. 87–92, New York, NY, USA, April 2021.

[74] P. Pradhan, "The role of arduino for increasing performance and interest in programming for first-year engineering students," Doctoral Dissertation, University of Cincinnati, Cincinnati, Ohio, 2017.

[75] S. Papadakis, "Is pair programming more effective than solo programming for secondary education novice programmers?" *International Journal of Web-Based Learning and Teaching Technologies*, vol. 13, no. 1, pp. 1–16, 2018.

[76] S. Dasuki and A. Quaye, "Undergraduate students' failure in programming courses in institutions of higher education in developing countries: a Nigerian perspective," *The Electronic Journal on Information Systems in Developing Countries*, vol. 76, no. 1, pp. 1–18, 2016.