International Journal of Sociologies and Anthropologies Science Reviews
Volume 5 Issue 4: July-August 2025: ISSN 2985-2730
Website: **https://so07.tci-thaijo.org/index.php/IJSASR/index**

The Impact of Blended Learning on Students' Grades in Introductory Programming Courses

Pinzhang Xie[1] and Lu Zhu[2]
[1] Ph.D. Candidate, Graduate School of Business and Advanced Technology Management, Assumption University, Thailand
[2] 2Program Director of Ph.D. Art, Music, Sports and Entertainment Management, Graduate School of Business and Advanced Technology Management, Assumption University, Thailand
[1]E-mail: 10958380@qq.com, ORCID ID: https://orcid.org/0009-0003-0065-504X
[2]Email: zhulu@au.edu, ORCID ID: https://orcid.org/0000-0001-6736-4309

Received 19/01/2025 Revised 23/01/2025 Accepted 23/02/2025

Abstract
Background and Aim: Blended learning has gradually become an important part of educational reform. This study selects first-year students majoring in Computer Science and Technology from Zhanjiang University of Science and Technology in China as the research subjects. It compares the application effects of blended learning and traditional teaching in introductory programming courses through a quasi-experimental design. The aim is to explore the impact of blended learning on students' programming course grades, verify its advantages over traditional teaching, and provide empirical references for computer programming education.
Materials and Methods: This study employed a quasi-experimental design, selecting 110 first-year Computer Science and Technology students. Participants were divided into experimental and control groups, each with 55 students. The experimental group used the Rain Classroom platform for blended learning, while the control group received traditional classroom instruction. Data on programming skills and course grades were collected for comparison.
Results: The findings revealed that there was no significant difference between the two groups in their understanding of programming concepts ($p = 0.058$, $d = 0.366$). Nevertheless, the experimental group demonstrated significantly higher performance than the control group in problem-solving skills ($p < .001$, $d = 1.636$), debugging and troubleshooting ($p < .001$, $d = 1.974$), and algorithmic thinking ($p < .001$, $d = 0.974$). Independent samples t-tests confirmed that blended learning significantly enhanced students' abilities in these practical skills and higher-order cognitive domains.
Conclusion: Compared with traditional teaching, blended learning has a significant effect on enhancing students' problem-solving skills, debugging and troubleshooting abilities, as well as algorithmic thinking. However, its direct impact on improving understanding of programming concepts may not be immediately evident. These findings provide strong evidence to support the application of blended learning in introductory programming courses and offer valuable references for future teaching practices and research.

Keywords: Blended Learning; Students' Grades; Impact; Introductory Programming Courses

Introduction

With the rapid development of technology and the comprehensive arrival of the digital age, information technology has penetrated all aspects of social life and has had a profound effect on the field of education (Hussaini, 2023). In this context, programming ability is gradually recognized as a core skill of the 21st century, which is not only related to personal career development but also an important manifestation of national competitiveness (Abesadze & Nozadze, 2020). Therefore, more and more countries and schools are incorporating programming into their curriculum to cultivate students' logical thinking, problem-solving, and innovation abilities (Moraiti et al., 2022). However, traditional Face-to-Face programming teaching methods are often limited by time, space, and teacher resources, making it difficult to meet the growing learning needs (Bhatti et al., 2019). Meanwhile, with the advancement of educational informatization and the rise of online education, the ways for students to acquire knowledge and skills have become more diversified and convenient (Politis & Politis, 2016). In this context, blended learning, as a new teaching model that combines the advantages of online learning and Face-to-Face teaching, has gradually gained widespread attention (Ananga & Biney, 2021). Blended learning not only breaks through the limitations of time and space, providing a more flexible and personalized learning experience, but also stimulates students' interest and creativity in learning through rich digital resources and interactive platforms. In the field of programming education, blended learning has unique advantages (Wang &Wong, 2012). Firstly, programming is a highly practical discipline that requires a lot of practice and feedback(Figueiredo& García,2022). Blended learning can provide students with more practical opportunities and immediate feedback, helping them better master programming skills (Demaidi et al., 2019). Secondly, programming is a subject that requires continuous learning and updating, and blended learning can provide students with ongoing learning support and resource updates, helping them keep up with the pace of technological development.

Objectives

This paper aims to explore the impact of blended learning on students' grades in introductory programming courses and attempts to achieve the following two research objectives:

1. Determine the differences in academic performance between two groups of students through two different teaching modes.

2. Determine the specific impacts on the academic performance of two groups of students through two different teaching modes.

Literature review

Blended learning

The blended learning theory emphasizes the combination of traditional face-to-face teaching and online learning to fully leverage the advantages of both teaching modes (Hrastinski, 2019). This theory focuses on how to effectively integrate different learning resources and activities to improve learning outcomes. This study is based on the theory of blended learning and explores the effect of blended learning on student performance by comparing the performance of traditional teaching groups and blended learning groups in introductory programming courses. This study not only validates the effectiveness of blended learning theory but also provides empirical support for its application in educational practice.

Understanding of Programming Concepts

Kristensen and Osterbye (1996) believe that understanding programming concepts involves comparing object-oriented languages from a conceptual perspective, considering concepts like phenomena and their abstract processes. Fundamentally, understanding programming concepts encompasses a basic understanding and cognition of programming, including core concepts such as variables, data types, functions, control structures (like conditional statements and loops), and object-oriented programming. These concepts are the cornerstone of programming and are essential for learning and mastering any programming language. Reynolds and Sverdlik (1995) suggest that understanding programming concepts is crucial for effectively retrieving software modules for reuse and can help learn classification rules for reuse libraries. Understanding programming concepts is vital for learning programming and maintaining existing programs, and visualization can enhance the understanding of programming and its underlying computer operations (Al-Fudaghi & Alashed, 2014). In the development process of programming, numerous researchers have conducted in-depth research on basic programming concepts, laying a solid foundation for the development of computer science. Tritrakan et al. (2016) proposed a blended learning environment model that significantly enhances students' understanding of programming concepts, their ability to solve problems using programming skills, and their program analysis skills.

Problem-Solving Skills

Dawkins et al. (2019) found that practicing problem-solving skills through programming can enhance student engagement. Hung (2008) pointed out that problem-solving skills teaching significantly improves the performance of computer engineering students in Verilog programming. Many scholars have studied how to improve students' problem-solving skills in programming and provided numerous reference methods. Malik et al. (2019) found that the use of PROBSOL applications improved the problem-solving ability of novice programmers, thereby increasing cognitive benefits and reducing the churn rate of introductory programming courses. Tu and Johnson (1990) found that successful completion of computer programming courses significantly enhances students' problem-solving abilities, specifically in the logical reasoning section. Gul et al.'s (2017) research shows that Mind Maps effectively improve students' problem-solving skills in both text and block-based programming languages. Sambe et al. (2021) found that integrating explicit guidance and semantic feedback systems in IDEs can promote problem-solving skills in computer programming.

Debugging and Troubleshooting

Li et al. (2019) assert that debugging is a crucial component of software development, yet novice programmers often struggle to formulate systematic strategies and processes for it. Michaeli and Romeike (2021) argue that debugging code is a fundamental skill in learning to program, with students' preexisting debugging experience being honed through troubleshooting, where they frequently encounter and resolve errors in their daily lives. Debugging is a systematic process aimed at discovering the reason why a given program does not function as intended and rectifying the underlying cause (Hromkovic & Staub, 2021). It is a process in computer programming and engineering where a problem is identified, isolated, corrected, and tested to ensure it operates as expected (Nita & Mihailescu, 2017).

Algorithmic Thinking

Algorithmic Thinking refers to the method of solving problems by following specific procedures or steps. This approach is invaluable in programming and problem-solving as it enables individuals to decompose complex problems into smaller, more manageable components, addressing them logically and systematically. According to Gonda et al. (2022), Algorithmic Thinking involves working with algorithms, understood as a systematic description of problem-solving strategies. Futschek (2006) emphasizes that Algorithmic Thinking is a core ability in IT that can be developed independently of programming skills and aids in understanding fundamental algorithmic concepts. Lamagna (2015) further adds that it encompasses the ability to understand, execute, evaluate, and create computational processes. Using Algorithmic Thinking as a metric to assess students' programming performance provides a comprehensive and deep understanding of their programming proficiency and problem-solving skills. This evaluation approach not only considers the accuracy of the code but also highlights the orderliness, logic, and innovation in the thinking process. Vinayakumar et al. (2018) stress that Algorithmic Thinking is vital for solving problems and enhancing comprehension of the computational process through programming and computation. Byrka et al. (2021) argue that higher education students must develop 21st-century skills and competencies in the information society. Gromova and Latanskaya (2021) reveal that research indicates Algorithmic Thinking is important for students in specialized computer science courses, and tasks based on fundamental algorithms can contribute to its development.

Conceptual Framework

The research framework, as the fundamental structure for research design and implementation, provides an important guiding framework for conducting research in an orderly manner and ensuring its effectiveness (Stevens & Finlay, 1996). Kaneko (2022) further emphasizes that a scientifically sound research framework should not only provide a solid foundation for the research process but also ensure that research problems can be effectively solved while ensuring that research results can be accurately and reasonably interpreted.

In this study, a rigorous research framework was designed using a quasi-experimental design approach. It should be noted that this experiment does not include the pre-test phase, only the post-test. Because the difference in total scores for first-year computer science majors is controlled within 10 points, and they have not received programming courses in high school, it can be reasonably assumed that their basic levels are equivalent, which provides a strong guarantee for the fairness of the experiment. 110 students were randomly divided into an experimental group and a control group, with 55 students in each group. The students in the experimental group participated in an introductory programming course using a blended learning mode, while the students in the control group participated in a traditional introductory programming course. The entire experimental cycle lasted for 8 weeks, during which the same course schedule and teaching progress control were applied to both groups of students.

At the end of the course, a post-test was conducted to comprehensively evaluate students' mastery in four key areas: Understanding of programming concepts (H1), problem-solving skills (H2), debugging and troubleshooting (H3), and algorithmic thinking (H4). Through this post-test, it is expected to verify the effectiveness of the blended learning mode in improving students' programming ability compared to the traditional learning mode.

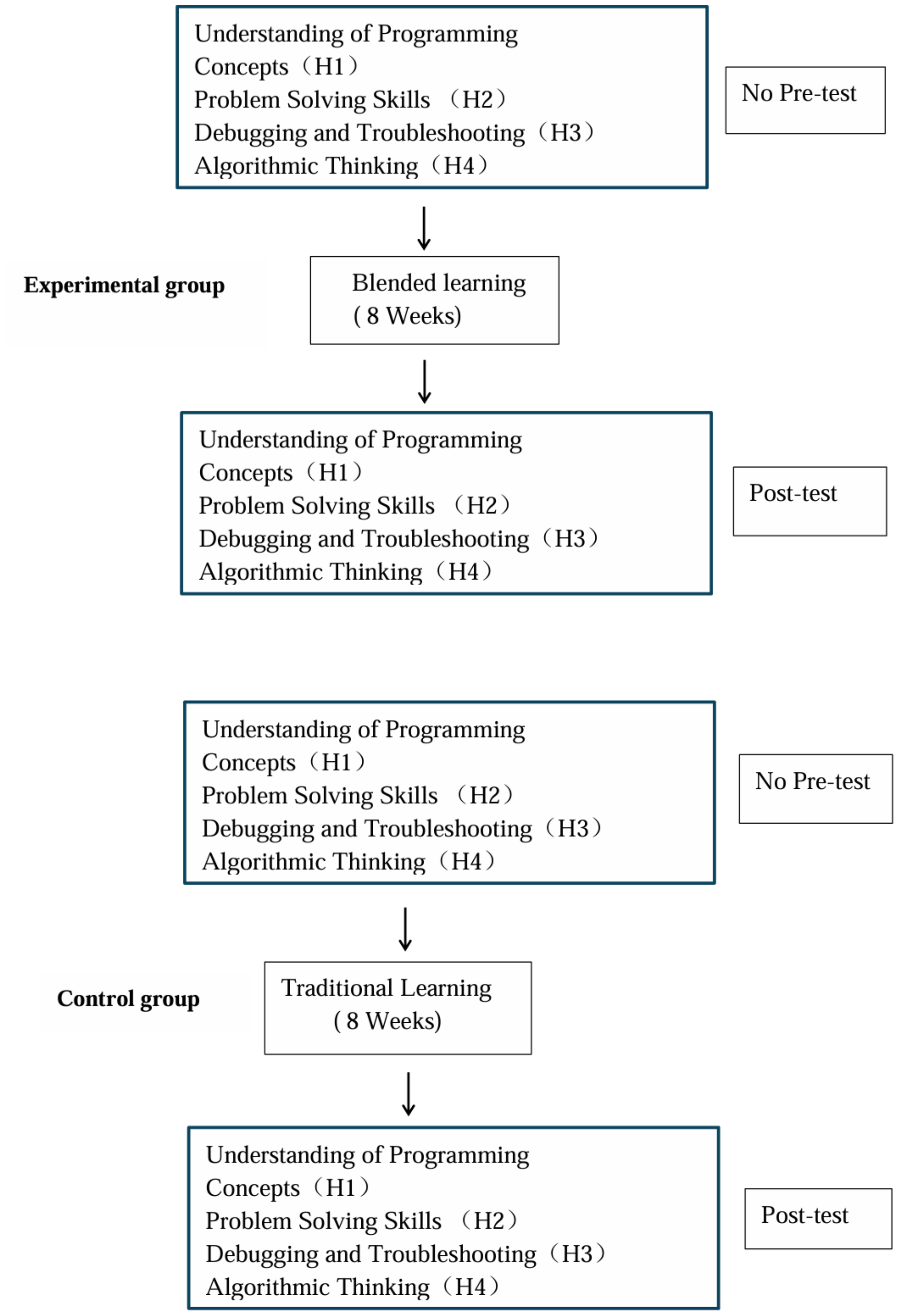

Understanding of Programming Concepts（H1）
Problem Solving Skills （H2）
Debugging and Troubleshooting（H3）
Algorithmic Thinking（H4）
No Pre-test
Experimental group
Blended learning (8 Weeks)
Understanding of Programming Concepts（H1）
Problem Solving Skills （H2）
Debugging and Troubleshooting（H3）
Algorithmic Thinking（H4）
Post-test
Understanding of Programming Concepts（H1）
Problem Solving Skills （H2）
Debugging and Troubleshooting（H3）
Algorithmic Thinking（H4）
No Pre-test
Control group
Traditional Learning (8 Weeks)
Understanding of Programming Concepts（H1）
Problem Solving Skills （H2）
Debugging and Troubleshooting（H3）
Algorithmic Thinking（H4）
Post-test

Figure 1 Research Framework

Research Hypothesis

Table 1 List of Hypotheses in the Study

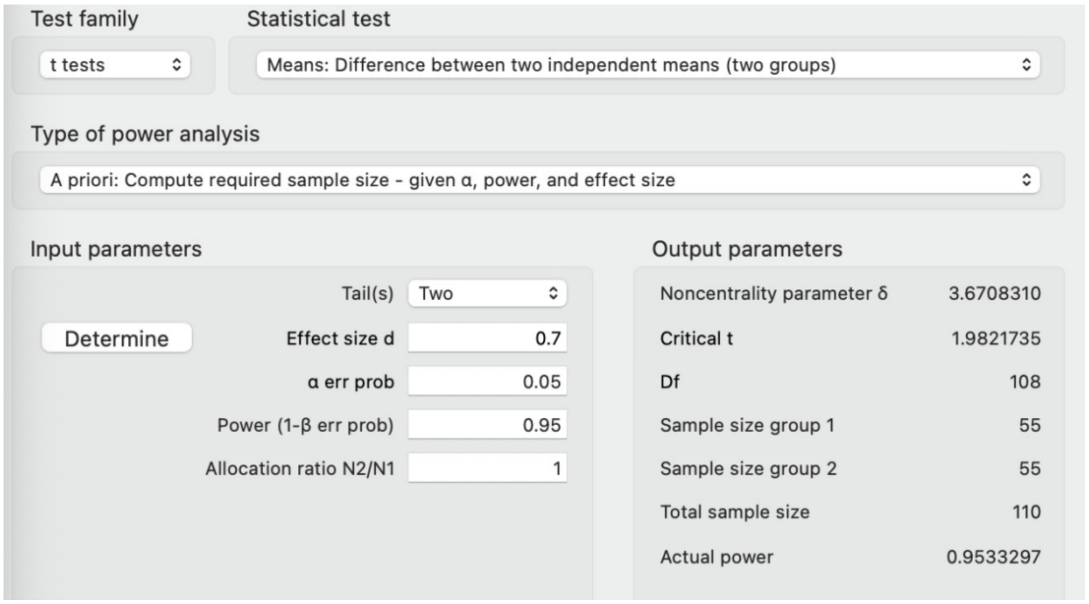
Hypotheses	Statement
H01	There is no significant difference in the understanding of programming concept s between the experimental group and the control group.
Ha1	There is a significant difference in the understanding of programming concepts between the experimental group and the control group.
H02	There is no significant difference in Problem-Solving Skills between the experimental group and the control group.
Ha2	There is a significant difference in Problem-Solving Skills between the experimental group and the control group.
H03	There is no significant difference in Debugging and Troubleshooting between the experimental group and the control group.
Ha3	There is a significant difference in Debugging and Troubleshooting between the experimental group and the control group.
H04	There is no significant difference in Algorithmic Thinking between the experimental group and the control group.
Ha4	There is a significant difference in Algorithmic Thinking between the experimental group and the control group.

Methodology

Population and sample

Population characteristics summarize the average and variance of individual genotypes or phenotypes influenced by various factors (Wade & McCauley, 1980). This study targets 240 Computer Science students at Zhanjiang University of Science and Technology during the first semester of the 2023-2024 academic year, comprising 96 females and 144 males. The chosen period captures students as they transition from high school, where C language programming is not part of the curriculum, to university, thereby providing a relatively uniform starting point for assessing their performance and adaptation in introductory programming courses. As a diverse institution that attracts students from across the country, Zhanjiang University of Science and Technology offers a rich research context, enhancing the generalizability of the findings. Given the common starting point in programming knowledge among the students, this study treats them as a relatively homogeneous group, minimizing the influence of individual differences on the results and enabling an accurate assessment of the effectiveness of blended learning models on their grades in programming courses.

In terms of sampling techniques, considering the consistency assumption of the gender ratio and knowledge level of the research subjects, this study will use a combination of stratified sampling techniques and random sampling techniques. The stratified sampling technique involves sampling from a population with mutually exclusive subgroups. Populations or strata, with the mean of each stratum being estimated by a random sample of observations from the population (Dalenius, 1950). The random sampling technique is a method for drawing samples without replacement from finite universes with unequal selection probabilities, allowing unbiased estimation of sampling variance (Horvitz & Thompson,1952). Specifically, students are first divided into two levels based on gender: female and male, and then randomly sampled within each level. This sampling method can ensure consistency in the gender ratio between the experimental group and the control group, thereby improving the internal validity of the study. Table 2 presents the sample used in this study. Use G * Power software for sample size estimation, which aims to provide accurate power analysis for most statistical tests in behavioral science (Faul et al., 2009). In this study, a priori analysis that computed the required sample size was calculated via the following values: a priori analysis for means difference between two independent means, Tail(s) 2, Effect size, 0.7 error probability of 0.05, 0.95 power, and Allocation ratio N2/N1. The results showed that each group required at least 55 samples for the study, with a total of 110 samples for both groups. As shown in Figure 2.


Test family: t tests
Statistical test: Means: Difference between two independent means (two groups)
Type of power analysis: A priori: Compute required sample size - given α, power, and effect size

Input parameters
Tail(s): Two
Effect size d: 0.7
α err prob: 0.05
Power (1-β err prob): 0.95
Allocation ratio N2/N1: 1

Output parameters
Noncentrality parameter δ: 3.6708310
Critical t: 1.9821735
Df: 108
Sample size group 1: 55
Sample size group 2: 55
Total sample size: 110
Actual power: 0.9533297

Figure 2 Sample size estimation using G * Power software

Table 2 List of Demographic Information of Samples

Groups	Category	Frequency	Percentage
Control group	Male	33	60%
	Female	22	40%
	Total	55	100%
Experimental group	Male	33	60%
	Female	22	40%
	Total	55	100%

Research Instruments

A research instrument is a tool used to collect, examine, investigate, and present data more systematically and objectively (Nida et al., 2022). Key considerations in selecting research instruments include reliability, validity, and sensitivity in research texts (Koop, 2003). The main research tool used in this study is testing.

Performance Tests

Pre-test and post-test control group designs are very suitable for investigating the effect of educational innovation and are common in educational research (Dugard &Todman, 1995). This study only had a post-test and no pre-test. After the experiment, the experimental group and the control group were tested to comprehensively evaluate students' learning outcomes in the introductory programming course, including an understanding of programming concepts, problem-solving skills, debugging, and troubleshooting, as well as proficiency in algorithmic thinking.

This study mainly focuses on two ways to learn introductory programming courses: the experimental group adopts a blended learning mode, while the control group uses traditional learning methods. The experimental group students fully utilized the resources of the Rain Classroom learning platform, including watching programming teaching videos, completing online exercises, participating in online discussions, answering questions, etc. The control group students followed traditional teaching methods, learning through classroom lectures, practice questions, and group discussions. Two groups of students maintain consistency in teaching content to ensure the effectiveness of the experiment. After the 8-week teaching period, a test will be conducted, and three computer programming experts will evaluate the test scores of the two groups. After the grading is organized, the score difference between the experimental group and the control group will be analyzed to evaluate the effectiveness of blended learning in improving students' course performance. In addition, based on the distribution and nature of the data, an independent sample t-test will be used for further data analysis. In short, this study evaluates the effectiveness of different teaching methods by comparing the grades of two groups of students.

Descriptive Statistics of Variables

In this study, we utilized Jamovi, a powerful data analysis software, to process and analyze the collected data. With its user-friendly interface and extensive statistical functions, jamovi has become our preferred tool for descriptive statistical analysis. For both the control and experimental groups in this study, we collected data from 55 students on four aspects: Understanding of programming concepts(UPC), problem-solving skills (PSS), Debugging and Troubleshooting (DT), and Algorithmic Thinking(AT). Using Jamovi, we calculated descriptive statistical indicators such as mean, standard deviation, and median for these variables, providing a comprehensive understanding of the distribution characteristics and variability of the data. Through the descriptive statistics tables and Q-Q plots generated by Jamovi, we can intuitively observe the performance differences among students in different groups across various variables, laying a solid foundation for subsequent data analysis and discussion of results. Next, we will conduct a detailed analysis and interpretation of these statistical data.

Table 3 Descriptive Statistics Table

	Group	**UPC**	**PSS**	**DT**	**AT**
N	Control group	55	55	55	55
	Experimental group	55	55	55	55
Mean	Control group	75.6	68.1	67.6	66.6
	Experimental group	78.8	82.4	81.7	75.0
Median	Control group	76	67	68	67
	Experimental group	78	83	82	76
Standard deviation	Control group	9.68	9.25	7.15	7.37
	Experimental group	7.24	8.16	7.17	9.62
Shapiro-Wilk W	Control group	0.969	0.927	0.979	0.971
	Experimental group	0.968	0.969	0.978	0.979
Shapiro-Wilk p	Control group	0.165	0.003	0.446	0.204
	Experimental group	0.149	0.161	0.407	0.461

Here is a detailed analysis of the descriptive statistics and normality determination for the four variables:

Understanding of Programming Concepts

Descriptive Statistics: The mean for the control group is 75.6, the median is 76, and the standard deviation is 9.68; the mean for the experimental group is 78.8, the median is 78, and the standard deviation is 7.24.

Analysis: The mean and median for the experimental group are both higher than those for the control group, indicating that the experimental group performs better in understanding programming concepts. Additionally, the standard deviation for the experimental group is relatively smaller, suggesting that the data distribution for the experimental group is more concentrated, with relatively consistent performance among students.

Normal Distribution: The results of the Shapiro-Wilk test show that the W value for the control group is 0.969 with a p-value of 0.165, and the W value for the experimental group is 0.968 with a p-value of 0.149. The p-values for both groups are greater than the commonly used significance level of 0.05; therefore, it can be considered that the data for both groups follow a normal distribution.

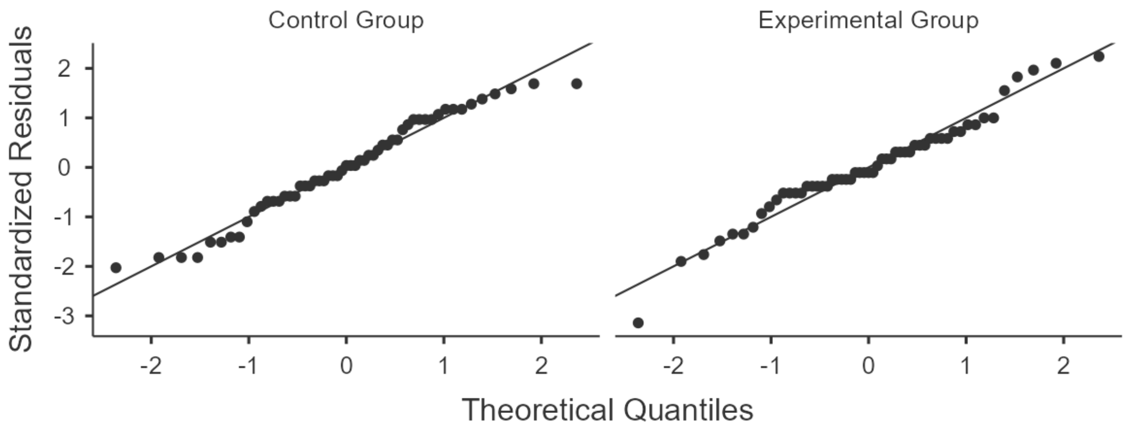


Figure 3 Analysis of Understanding of Programming Concepts Variable using Q-Q Plots

Problem-Solving Skills

Descriptive Statistics: The mean for the control group is 68.1, the median is 67, and the standard deviation is 9.25; the mean for the experimental group is 82.4, the median is 83, and the standard deviation is 8.16.

Analysis: Both the mean and median for the experimental group are significantly higher than those for the control group, indicating that the experimental group performs markedly better in problem-solving ability. The standard deviation for the experimental group is relatively smaller, suggesting that the data distribution for the experimental group is more concentrated, with relatively consistent performance among students in problem-solving ability.

Normal Distribution: The results of the Shapiro-Wilk test show that the W value for the control group is 0.927 with a p-value of 0.003, and the W value for the experimental group is 0.969 with a p-value of 0.161. Although the p-value for the control group is less than 0.05, considering the relatively large sample size and the result, along with the p-value for the experimental group being greater than 0.05, we can cautiously conclude that the data for both groups follow a normal distribution overall.

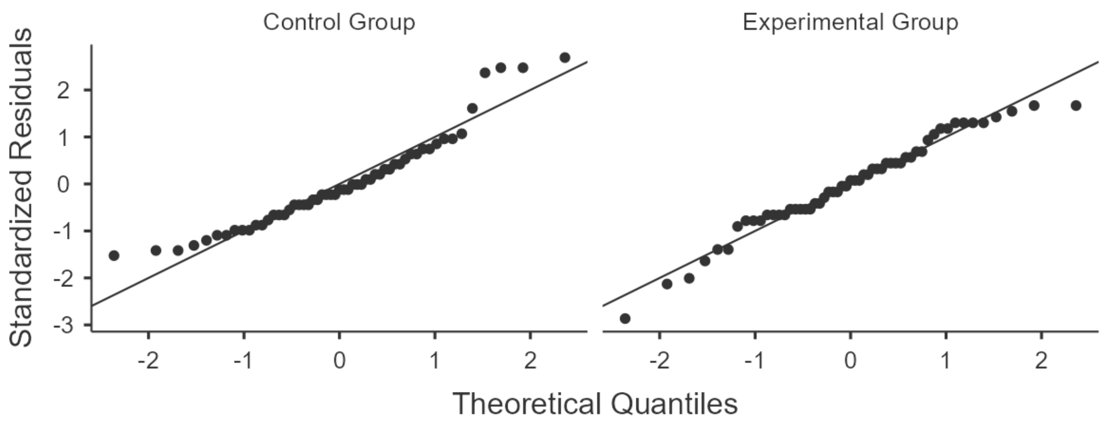


Figure 4 Analysis of problem-solving skills Variable using Q-Q Plots

Debugging and Troubleshooting

Descriptive Statistics: The mean for the control group is 67.6, the median is 68, and the standard deviation is 7.15; the mean for the experimental group is 81.7, the median is 82, and the standard deviation is 7.17.

Analysis: The mean and median for the experimental group are both higher than those for the control group, indicating that the experimental group performs better in debugging and troubleshooting. The standard deviations for both groups are similar, suggesting that the data distributions for both sets of students are relatively concentrated.

Normal Distribution: The results of the Shapiro-Wilk test show that the W value for the control group is 0.979 with a p-value of 0.446, and the W value for the experimental group is 0.978 with a p-value of 0.407. The p-values for both groups are greater than the commonly used significance level of 0.05; therefore, it can be considered that the data for both groups follow a normal distribution.

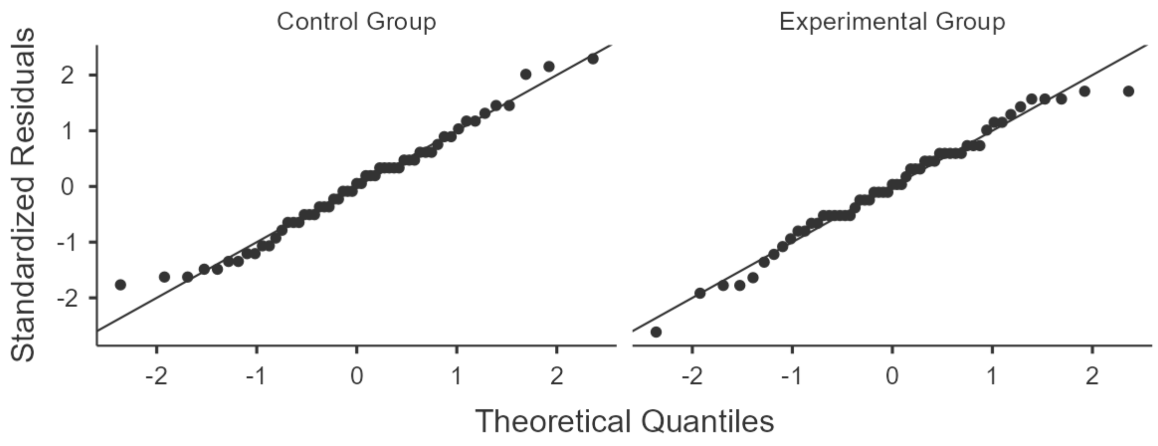


Figure 5 Analysis of Debugging and Troubleshooting Variables using Q-Q Plots

Algorithmic Thinking

Descriptive Statistics: The mean for the control group is 66.6, the median is 67, and the standard deviation is 7.37; the mean for the experimental group is 75.0, the median is 76, and the standard deviation is 9.62.

Analysis: The mean and median for the experimental group are both higher than those for the control group, indicating that the experimental group performs better in algorithmic thinking. However, the standard deviation for the experimental group is relatively larger, suggesting that the data distribution for the experimental group is more dispersed, with some variation in students' performance in algorithmic thinking.

Normal Distribution: The results of the Shapiro-Wilk test show that the W value for the control group is 0.971 with a p-value of 0.204, and the W value for the experimental group is 0.979 with a p-value of 0.161. The p-values for both groups are greater than the commonly used significance level of 0.05; therefore, it can be considered that the data for both groups follow a normal distribution.

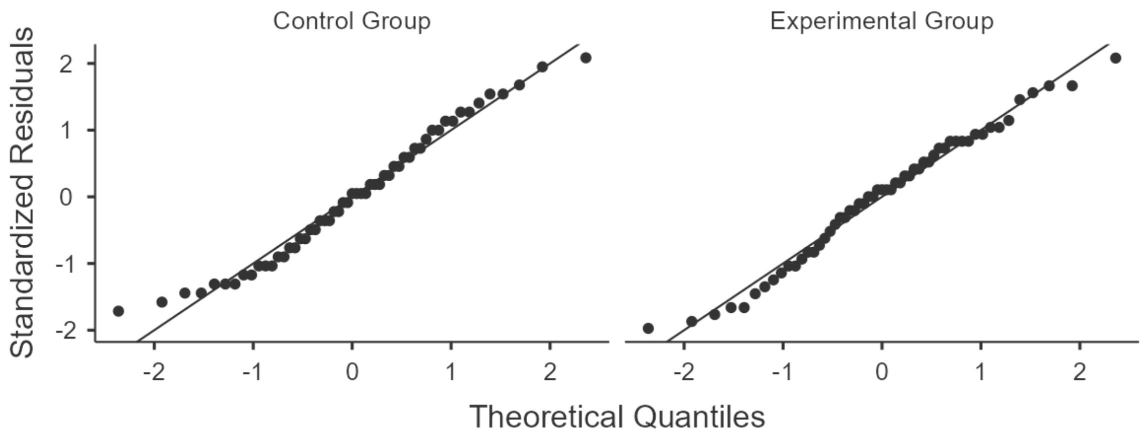


Figure 6 Analysis of Algorithmic Thinking Variable using Q-Q Plots

Results

In this study, an Independent Samples T-Test was utilized to investigate the performance differences in four distinct measurement indicators (understanding of programming concepts, problem-solving skills, debugging and troubleshooting, and algorithmic thinking) between the experimental group and the control group of students.

Table 4 Independent Samples T-Test

		Statistic	df	p	Mean difference	SE difference	Effect Size
UPC	Student's t	1.92	108	0.058	3.13	1.63	0.366
PSS	Student's t	8.58	108	< .001	14.27	1.66	1.636
DT	Student's t	10.35	108	< .001	14.13	1.36	1.974
AT	Student's t	5.11	108	< .001	8.35	1.63	0.974

As shown in Table 4, the following results were obtained:

The findings revealed no significant difference between the two groups in their understanding of programming concepts (p = 0.058, d = 0.366). Nevertheless, the experimental group demonstrated significantly higher performance than the control group in problem-solving skills (p < .001, d = 1.636), debugging and troubleshooting (p < .001, d = 1.974), and algorithmic thinking (p < .001, d = 0.974). Independent samples t-tests confirmed that blended learning significantly improved students' abilities in these practical skills and higher-order cognitive domains. This study suggests that blended learning holds considerable potential in introductory programming education, especially in fostering practical skills and higher-order thinking abilities. These findings offer practical insights for the reform of programming courses in higher education institutions. Table 5 presents a summary of the results from hypothesis testing in the study.

Table 5 Summary of Hypothesis testing and results

Hypotheses	Statement	Result after Analysis
H01	There is no significant difference in the An understanding of programming concepts between The experimental group and the control group	Accept, There is no significant difference
H02	There is no significant difference in Problem-Solving Skills between the experimental group and the control group.	Reject, There exists a significant difference
H03	There is no significant difference in Debugging and Troubleshooting between the experimental group and the control group.	Reject, There exists a significant difference
H04	There is no significant difference in Algorithmic Thinking between the experimental group and the control group.	Reject, There exists a significant difference

Discussion

The research results indicate that students in the experimental group scored higher on the post-tests than those in the control group. This finding suggests that blended learning methods are beneficial for improving the academic performance of students in introductory programming courses. The results of this study partially confirm the findings of Tritrakan et al. (2016) that a blended learning environment significantly improves students' ability to solve problems using programming skills, but no significant difference was found in enhancing students' Understanding of Programming Concepts. Meanwhile, this study also validates the viewpoint proposed by Lin et al. (2021) that a blended learning environment promotes computational thinking.

Advantages Analysis

a) Prominent Advantages of Blended Learning: The main finding of this study is that blended learning demonstrates significant advantages over traditional teaching in enhancing students' problem-solving skills, debugging and troubleshooting abilities, as well as algorithmic thinking. This aligns with existing research, such as the study by Tritrakan et al. (2016), emphasizing the importance of blended learning in cultivating students' practical skills.

b) Unexpected Gains in Algorithmic Thinking: We unexpectedly found that students in the experimental group excelled particularly in algorithmic thinking. This was not only reflected in their post-test scores but also in their enhanced logical thinking and problem-solving capabilities, which were evident in both class discussions and after-class interactions. This suggests that blended learning may subtly promote higher-order development of students' computational thinking.

c) Improvement in Learning Attitude and Participation: Students in the experimental group exhibited higher learning enthusiasm and participation, engaging more actively in class discussions, completing assignments more diligently, and proactively exploring additional learning resources and challenges. This could be attributed to the flexibility and autonomy of the blended learning model, which provides students with a more personalized and interactive learning experience, thereby stimulating their learning motivation.

Disadvantages Analysis

a) Inadequate Understanding of Programming Concepts: Consistent with the study by Tritrakan et al. (2016), this study did not find significant advantages of blended learning in improving students' understanding of programming concepts. This may indicate that blended learning may not be as effective as traditional teaching in imparting theoretical knowledge. Further exploration is needed to optimize the design of blended learning to address this shortcoming. For example, considerations could include adding online explanations, concept discrimination exercises, and targeted practice sessions.

b) Short Study Duration: The teaching intervention in this study lasted for 8 weeks, which may be relatively short and insufficient to fully assess the long-term impact of blended learning. To more comprehensively evaluate the effectiveness of blended learning, future studies should adopt longer-term tracking surveys.

c) Limited Sample Selection: The sample in this study was limited to computer science and technology students from Zhanjiang Science and Technology University, which may limit the representativeness of the sample. The research results may differ in other contexts and types of schools. Future studies could consider expanding the sample range to include students from different schools and majors to enhance the generalizability of the research findings.

Conclusion

The study conducted an in-depth comparison of the implementation effects of blended learning and traditional teaching modes in introductory programming courses, and drew the following conclusions: In the stage of teaching basic programming concepts, traditional classroom teaching or blended learning modes can continue to be used to ensure that students lay a solid foundation; whereas, in cultivating students' problem-solving abilities, debugging and troubleshooting as well as Algorithmic Thinking and other advanced programming skills, a blended learning mode should be preferred. This mode combines autonomous learning on an online platform with in-depth instruction in offline classes, providing students with more opportunities for practice and interaction, effectively stimulating their interest and enthusiasm for learning.

This research result not only brings a new perspective to programming education practice but also has far-reaching implications. Blended learning mode, with its rich and diverse teaching resources and flexible learning methods, fully respects students' differences and learning needs, opening up a new path for programming instruction. Therefore, we should actively adopt and promote the blended learning mode, optimizing the teaching design of programming courses to achieve better teaching effects. In the wave of digitalization, programming skills have become an important indicator of an individual's comprehensive quality, and the teaching quality and learning outcomes of introductory programming courses are directly related to students' future career development. Through empirical analysis, this study reveals the significant role of blended learning in improving students' programming performance, providing strong theoretical support and practical guidance for the teaching reform of programming courses. At the same time, the results of this study also offer useful references and inspirations for the teaching reform of other disciplines. The successful application of the blended learning mode not only injects new vitality into programming education but also provides new ideas and methods for the information-based teaching of other disciplines. By continuously optimizing the teaching design and learning support services in the blended learning environment, we can better meet students' learning needs, improve their learning outcomes and interests, and thus promote the comprehensive development of education informatization.

Recommendation

In summary, this study has revealed that blended learning offers notable benefits in enhancing students' problem-solving skills, debugging and troubleshooting, and algorithm thinking, although its immediate impact on improving understanding of programming concepts may not be evident. Based on these findings, we suggest promoting the blended learning approach in introductory programming courses, as it combines the strengths of online and face-to-face instruction to offer a more enriched, flexible, and tailored learning experience. Additionally, we recommend exploring more effective teaching strategies and methods to address the challenge of understanding programming concepts, to comprehensively enhance students' programming proficiency and overall literacy. Future research should consider extending the experimental period, increasing the sample size, and incorporating a broader range of evaluation indicators and methods to gain a more comprehensive understanding of the long-term effects of blended learning on students' programming skills and overall competence.

References

Abesadze, S., & Nozadze, D. (2020). Make 21st century education: The importance of teaching programming in schools. *International Journal of Learning and Teaching, 6*(3), 158-163.

Al-Fedaghi, S., & Alrashed, A. (2014). Visualization of Execution of Programming Statements. 2014 11th International Conference on Information Technology: *New Generations*, 363-370.

Ananga, P., & Biney, I. (2021). Comparing Face-To-Face And Online Teaching And Learning In Higher Education. *MIER Journal of Educational Studies, Trends and Practices, 7*, 165-179.

Bhatti, S., Dewani, A., Maqbool, S., & Memon, M. A. (2019). A Web-based Approach for Teaching and Learning Programming Concepts at Middle School Level. *International Journal of Modern Education & Computer Science, 11*(4), 46-53. DOI: 10.5815/ijmecs.2019.04.06

Byrka, M., Sushchenko, A., Svatiev, A., Mazin, V., & Veritov, O. (2021). A New Dimension of Learning in Higher Education: Algorithmic Thinking. *Propósitos y Representaciones, 9 (2)*, 990. https://doi.org/10.20511/pyr2021.v9nSPE2.990

Dalenius, T. (1950). The Problem of Optimum Stratification. *Scandinavian Actuarial Journal, 1950*, 203-213.

Dawkins, H., Gillis, D., & McCuaig, J. (2020). Validation of an expert problem-solving behavior scale for computer science education. In *ICERI2020 Proceedings* (pp. 6755-6764). IATED.

Demaidi, M., Qamhieh, M., & Afeefi, A. (2019). Applying Blended Learning in Programming Courses. *IEEE Access, 7*, 156824-156833.

Dugard, P., & Todman, J. (1995). Analysis of Pre-test-Post-test Control Group Designs in Educational Research. *Educational Psychology, 15*, 181-198.

Faul, F., Erdfelder, E., Buchner, A., & Lang, A. G. (2009). Statistical power analyses using G* Power 3.1: Tests for correlation and regression analyses. *Behavior research methods, 41*(4), 1149-1160.

Figueiredo, J., & García-Peñalvo, F. (2022). Strategies to increase success in learning programming. *2022 International Symposium on Computers in Education (SIIE)*, 1-6. https://doi.org/10.1007/978-981-97-1814-6_15

Futschek, G. (2006, November). Algorithmic thinking: the key to understanding computer science. *In International conference on informatics in secondary schools-evolution and perspectives* (pp. 159-168). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gonda, D., Ďuriš, V., Tirpáková, A., & Pavlovičová, G. (2022). Teaching algorithms to develop the algorithmic thinking of informatics students. *Mathematics, 10*(20), 3857.

Gromova, S. F., & Latanskaya, I. V. (2021, July). Basic algorithms as a means of developing algorithmic thinking in students when learning a specialized computer science course. *In 9th International Scientific & Practical Conference "Culture, Science, Education: Problems and Perspectives*, 1 (1), 477-485.

Gul, S., Asif, M., Ahmad, W., & Ahmad, U. (2017). Teaching programming: A mind map-based methodology to improve learning outcomes. *2017 International Conference on Information and Communication Technologies (ICICT),* 209-213.

Horvitz, D., & Thompson, D. (1952). A Generalization of Sampling Without Replacement from a Finite Universe. *Journal of the American Statistical Association*, *47*, 663-685.

Hrastinski, S. (2019). What do we mean by blended learning? *TechTrends, 63*(5), 564-569.

Hromkovic, J., & Staub, J. (2021). The problem with debugging in current block-based programming environments is. *Bulletin of EATCS, 135*(3),1-12.

Hung, Y. (2008). The Effect of Problem-Solving Instruction on Computer Engineering Majors' Performance in Verilog Programming. *IEEE Transactions on Education*, *51*, 131-137.

Hussaini, M. H. A. (2023). Effect of Information Technology on Education. *Graduate Journal of Pakistan Review (GJPR)*, 3(1).https://www.pakistanreview.com/index.php/GJPR/article/view/134

Kaneko, Y. (2022). Introduction of Research Framework. *SpringerBriefs in Economics*, 1–19.

Kanselaar G. 2002. Constructivism and socio-constructivism. *Article published on July 16*, 2002.

Koop, P. (2003). Finding and evaluating potential research instruments. *Canadian oncology nursing journal = Revue canadienne de nursing oncologique*, 13(4), 207-208 .

Kristensen, B., & Østerbye, K. (1996). A conceptual perspective on the comparison of object-oriented programming languages. *ACM SIGPLAN Notices*, *31*, 42-54.

Lamagna, E. (2015). Algorithmic thinking unplugged. *Journal of Computing Sciences in Colleges, 30*, 45-52.

Li, C., Chan, E., Denny, P., Luxton-Reilly, A., & Tempero, E. (2019). *Towards a Framework for Teaching Debugging.* 2019, 79-86.

Lin, X., Y., W., Liu, Y., & Tang, W. (2021). Using peer code review to improve computational thinking in a blended learning environment: A randomized control trial. Computer Applications in Engineering Education, 29, 1825 - 1835.

Malik, S., Mathew, R., Al-Nuaimi, R., Al-Sideiri, A., & Coldwell-Neilson, J. (2019). Learning problem-solving skills: Comparison of E-learning and M-learning in an introductory programming course. *Education and Information Technologies*, 1-18. DOI:10.1007/s10639-019-09896-1

Michaeli, T., & Romeike, R. (2021). Developing a Real World Escape Room for Assessing Preexisting Debugging Experience of K12 Students. *2021 IEEE Global Engineering Education Conference (EDUCON)*, 521-529.

Moraiti, I., Fotoglou, A., & Drigas, A. (2022). Coding with Block Programming Languages in Educational Robotics and Mobiles, Improved Problem Solving, Creativity & Critical Thinking Skills. *Int. J. Interact. Mob. Technol.*, *16*, 59-78.

Nida, N. F., Fauzie, M. M., & Istiqomah, S. H. (2021). Instrumentasi Pemeriksaan Sanitasi Pada Pembuatan Jamu Skala Industri Rumah Tangga. *Sanitasi: Jurnal Kesehatan Lingkungan*, *14(*2), 92-99.

Nita, S. L., Mihailescu, M., Nita, S. L., & Mihailescu, M. (2017). Interactive Debugger for Development and Portability Applications Based on Big Data. *Practical Concurrent Haskell: With Big Data Applications*, 221-230.

Politis, J., & Politis, D. (2016). The Relationship Between an Online Synchronous Learning Environment and Knowledge Acquisition Skills and Traits: The Blackboard Collaborate Experience. *Electronic Journal of e-Learning*, *14*, 196-203.

Reynolds, R., & Sverdlik, W. (1995). An Evolution-Based Approach to Program Understanding Using Cultural Algorithms. *Int. J. Softw. Eng. Knowl. Eng.*, *5*, 211-226.

Sambe, G., Drame, K., & Basse, A. (2021). Towards a Framework to Scaffold Problem-solving Skills in Learning Computer Programming. *In CSEDU (1)* (pp. 323-330).

Stevens, C. A., & Finlay, P. N. (1996). A Research Framework for Group Support Systems. *Springer EBooks*, 221–243.

Tritrakan, K., Kidrakarn, P., & Asanok, M. (2016). The Use of Engineering Design Concept for Computer Programming Course: A Model of Blended Learning Environment. *Educational Research Review, 11*, 1757-1765.

Tu, J., & Johnson, J. (1990). Can computer programming improve problem-solving ability? *Acm Sigcse Bull., 22*, 30-33.

Vinayakumar, R., Soman, K., & Menon, P. (2018). Alg-Design: Facilitates Learn Algorithmic Thinking for Beginners. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 1-6.

Wade, M. J., & McCauley, D. E. (1980). Group selection: The phenotypic and genotypic differentiation of small populations. *Evolution*, 34(4), 799-812.https://doi.org/10.1111/j.1558-5646.1980.tb04019.x

Wang, F. L., & Wong, T. L. (2010). Hybrid Teaching and Learning of Computer Programming Language. In *Handbook of Research on Hybrid Learning Models: Advanced Tools, Technologies, and Applications* (pp. 487-502). IGI Global.