engineering proceedings

MDPI

Proceeding Paper

Dynamic Program Analysis and Visualized Learning System in University Programming Courses †

Pei-Wen Lin [1,*], Shu-Han Yu [1] and Chien-Hung Lai [2]

[1] Undergraduate Program in Electrical Engineering & Computer Science, Chung Yuan Christian University, Taoyuan 320314, Taiwan; kellyyu024@gmail.com
[2] Department of Information and Computer Engineering, Chung Yuan Christian University, Taoyuan 320314, Taiwan; soulwind@cycu.org.tw
* Correspondence: dollylin921101@gmail.com
† Presented at the 2024 4th International Conference on Social Sciences and Intelligence Management (SSIM 2024), Taichung, Taiwan, 20–22 December 2024.

Abstract

To correspond to the advancement of technology, programming has become an indispensable course in university curricula. However, students easily become confused by the rules governing program execution or by complex logical structures. Mastering program structure and logic often is a significant challenge for beginners, especially. Despite the availability of information on programming on various websites and tools, including generative artificial intelligence (AI), there is still a gap between conceptual understanding and practical application for beginners. They overlook important implementation details or struggle to grasp the flow of a program, making the mastery of program logic a persistent challenge. To address these issues, we have developed a system that dynamically generates process architecture diagrams. Users upload their code, and the system produces corresponding diagrams that decompose and execute the code line by line. Its visual representation allows users to observe the program's execution and aids them in comprehending the sequence and operational flow of the code. By understanding the structure and logic of the program intuitively, this system supplements traditional teaching methods and AI-assisted question-and-answer tools. The experimental results demonstrated that students found the system helpful to track their learning progress (87%) and improved their understanding of program code (81%). Additionally, 84% of students reported that the system was easy to use, highlighting its user-friendliness. In terms of student interest, 83% of students agreed that the interactive elements made learning more engaging, indicating that the system positively contributed to dynamic and enjoyable learning. However, 63% of students reported an improvement in coding and were influenced by the complexity of the programming tasks assigned. Despite this, the overall satisfaction with the system developed in this study was high.

Keywords: programming learning; dynamic flowchart; visualization learning; program analysis

check for updates

Academic Editors: Teen-Hang Meen, Liza Lee and Cheng-Fu Yang

Published: 2 July 2025

Citation: Lin, P.-W.; Yu, S.-H.; Lai, C.-H. Dynamic Program Analysis and Visualized Learning System in University Programming Courses. *Eng. Proc.* **2025**, *98*, 30. https://doi.org/10.3390/engproc2025098030

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

College students feel that programming is difficult as it is not easy for them to imagine or understand the execution process of the code and the internal operation in the execution of the program. Even if professors adopt various teaching strategies to help students master these concepts, most students cannot understand the details of program execution.

For example, to understand basic concepts such as variables, operations, conditional judgments, loops, and function calls in C programming language, students spend a lot of time. In courses teaching advanced data forms such as arrays, structures, and linked lists, the program structure becomes more complex, and it is more difficult for students to understand its operation process, which eventually leads some students to abandon learning programming. Therefore, it is necessary to effectively help students understand the running process of the program and master its working principles.

To overcome such challenges, teaching strategies, such as using teaching tools to assist in explanation, have been developed [1], using flipped teaching [2], educational Chatbots to enhance programming [3], and computer programs to assist teaching [4].

In this study, we developed a visualization teaching method to enhance students' understanding of program logic and to learn and understand the program structure effectively [5]. It is easier to present abstract concepts graphically than in plain text. Therefore, we applied visualization technology to teach programming and help students understand program concepts more deeply. The developed system dynamically generates flowcharts and architecture diagrams. By graphically presenting the program architecture, students can see the operation process and structure of the program reducing the difficulty of learning and improving the learning effect.

2. Literature Review

2.1. Visualization System

In practicing programming, the abstract concepts of the program need to be imagined in the mind, which is difficult for beginners to understand. Many beginners refer to textbooks or examples on the Internet to learn existing programs to understand their logic in group discussion. They can understand the program fragments but often do not see the role of the overall program operation [6]. The wholeness and organization of human beings in perception and learning are linked to the cognition of graphics.

The key to solving the problem is to understand the overall structure of the program, rather than simply combining parts [7], which aligns with the design concept of the program visualization system. The visualization tool displays the program structure and processes it through images, which helps beginners understand program logic and concepts. It also allows them to easily view program blocks as a whole and understand the application. Additionally, it encourages students to perceive the context of program operation step by step [8].

2.2. Program Visualization

Programming language is difficult for beginners to learn, as language diversity, grammar rules, program architecture, abstract data structure, functions, and other concepts must be understood. Beginners can expand their understanding of programming languages step by step through the learning process and practice these concepts. However, it often takes time for beginners to fully apply the concepts. In addition, the lack of problem-solving skills is also an obstacle. Solving programming problems involves four steps, such as "understanding the problem", "designing the framework", "writing code", and "testing and debugging". For beginners, decomposing problems, making plans, and implementing those plans through programming languages to solve problems is challenging [9]. Previous studies indicated that students cannot understand and break down problems. This results in an inability to solve issues or organize solutions into a cohesive plan. To address these difficulties and help beginners accelerate their mastery of programming, the visual learning environment is used as an effective teaching tool. It helps students lower the learning threshold and increase their interest in programming [10].

3. Experimental Methods

We have developed a system that dynamically generates flowcharts and architecture diagrams "process visualization system" (Figure 1). The system accurately extracts the necessary information from the code and converts it into a visualized flowchart and architecture diagram. After the system was designed, students evaluated its effectiveness in programming learning. The experiment was conducted to collect feedback from students after using the system through a questionnaire survey to analyze the impact of the system on students' learning.

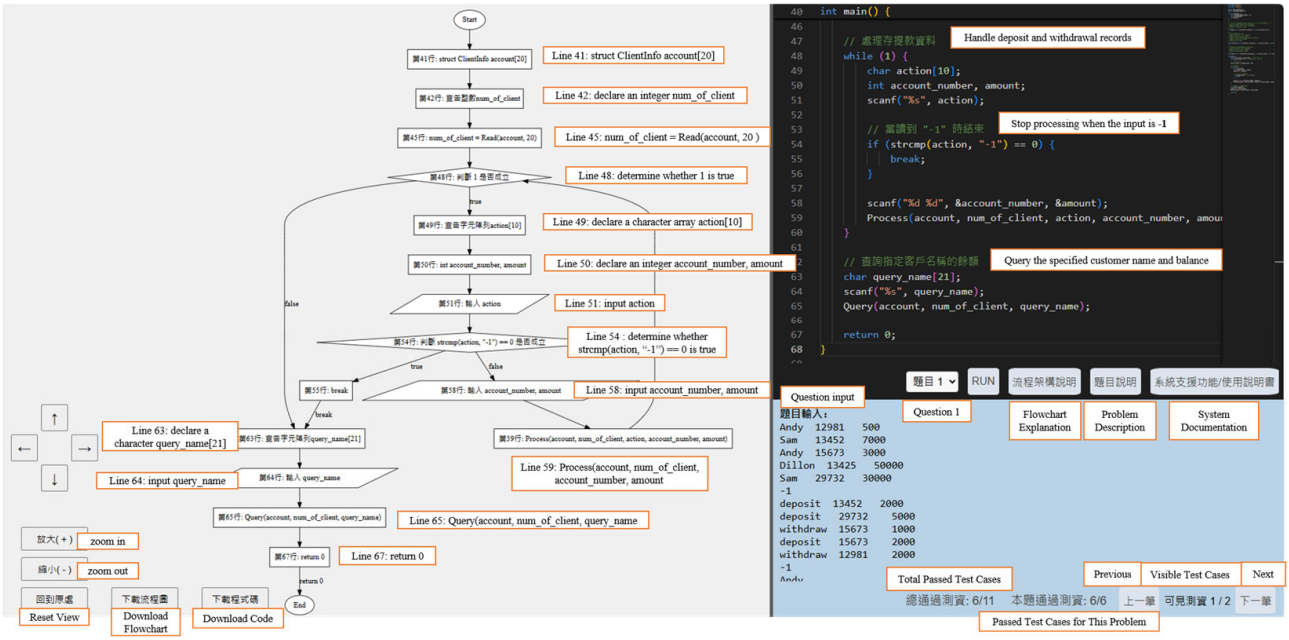


Figure 1. System diagram.

3.1. Subject and Course

First-year university students in northern Taiwan participated in this study. All participants majored in electrical engineering and computer science and did not have prior programming knowledge. Before the experiment, they were introduced to basic programming concepts. The students did not understand the structure and processes of the program operation. After using the developed system, the students evaluated the effect of visualization tools on programming learning.

A course "Introduction to Computers" was offered to teach C programming. Basic and advanced concepts, including variables, operations, conditional judgment, loops, function calls, arrays, structures, and linked lists were taught. The course content included the foundation of programming, which was ideal for testing the effectiveness of the developed visualization tools for the first-year students.

3.2. Process

The experiment was conducted in two phases (Figure 2). Before the experiment, the professor explained the concepts to help students understand the basic structure and operating logic of the C language and have basic programming knowledge. The first test was conducted one week after the course began without using the visualization tool to evaluate students' basic understanding of the content. The second test was conducted in the following week after using the developed tool. This allowed the students to use the system in programming tasks. After the experiment, students filled out a questionnaire on their experience with the tool and its learning effect.

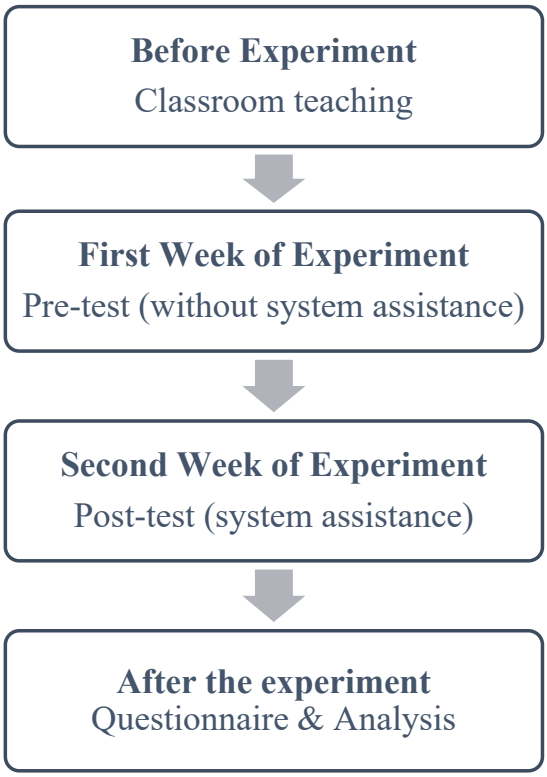


Figure 2. Experimental process.

4. Results and Discussion

The questionnaire survey data were analyzed to explore the impact of the developed tool that generated flowcharts showing students' learning effects and motivation. Based on the survey results and data analysis, the usability of the tool, improvement in learning motivation, the impact of peer competition, and learning performance were evaluated.

4.1. System Usability

In total, 84% of the students found the tool easy to use. This indicated that the tool's interface was intuitive, allowing users to quickly understand how to operate it and learn after familiarizing themselves with it (Table 1). The participants showed their expectations for additional system features. We improved several tool functions to make the learning experience more comfortable.

Table 1. Response to Question "I think the tool is easy to use".

Strongly Disagree	3.23%
Disagree	12.90%
Neutral	32.26%
Agree	32.26%
Strongly Agree	19.35%

4.2. Learning Effect

4.2.1. Learning Progress

In total, 87% of the students believed that the tool enhanced their ability to track learning progress(Table 2). They thought that the tool helped them change their perception of program mastery in the learning process, understand learning status, and improve their programming skills.

Table 2. Response to Question "The tool enhances students' ability to track their learning progress".

Strongly Disagree	3.23%
Disagree	9.68%
Neutral	25.81%
Agree	48.39%
Strongly Agree	12.90%

4.2.2. Program Ability

Totally, 81% of the students said that the tools improve their understanding of the code (Table 3). The tool converted students' code into flowcharts, presenting program logic concisely and intuitively, making it easier for students to understand. This aided them in learning program logic. Through graphical presentation, students understand the course content, thus improving their learning outcomes.

Table 3. Response to Question "The tool is helpful in improving program understanding".

Strongly Disagree	0.00%
Disagree	19.35%
Neutral	54.84%
Agree	19.35%
Strongly Agree	6.45%

4.2.3. Interest and Confidence in Learning

In total, 83% of the students believed that interactive elements made learning attractive (Table 4). These interactive elements allowed the students to see the results of program execution in real time but also dynamically displayed the corresponding blocks of the flowchart to each line of code.

Table 4. Response to Question "System interaction elements make learning more attractive".

Strongly Disagree	3.33%
Disagree	13.33%
Neutral	30.00%
Agree	43.33%
Strongly Agree	10.00%

4.2.4. Confidence in Learning

In total, 63% of the students said that their confidence in learning programming in the course improved (Table 5). While the visualization tool effectively assisted students in understanding the program's running process, several students felt less confident. In the programming-learning process, the tool did not remove the difficulties of learning that the students felt.

Table 5. Response to Question "The tool helps the confidence of programming".

Strongly Disagree	10.00%
Disagree	26.67%
Neutral	36.67%
Agree	20.00%
Strongly Agree	6.67%

5. Conclusions

We have developed a visualization tool that dynamically generates flowcharts based on the code and applies it in teaching, utilizing a graphical teaching method. The tool was applied in a programming language course to improve student learning outcomes.

The tool helped students understand programming language logic by converting code into flowcharts. Compared with traditional learning methods, the tool enabled students to quickly grasp the program's execution process. By observing the flowchart, the students understood the relationship between different steps, thus improving their mastery of programming languages.

The visual learning tool converted abstract code into concrete flowcharts, allowing students to intuitively understand the logic behind program execution. The visualization of flowcharts made the learning process intuitive, helping students reduce psychological barriers when learning complex program structures. This learning method enhanced students' acceptance of programming languages and improved their ability to master programming concepts, further stimulating their interest in programming.

Students' learning mentality—such as their confidence in solving programming problems—also affects their progress. Therefore, the tool needs to be enhanced by adopting AI. AI models allow students to ask questions and engage in continuous dialog, adjusting the teaching method based on each student's learning process. For instance, in learning programming architecture, AI assists students in breaking down problems and provides tailored guidance according to their thinking patterns.

The flowchart can be enhanced in an animated format to show the program execution process in more detail, providing a clearer view of how the program operates. By adding more detailed visual representations, the flowchart improves its visual appeal and boosts students' interest and engagement in the learning process. The tool needs to incorporate a variable tracking feature to adjust the program's execution interval, observe changes in the program's variables, and increase the interactivity with students. Such functionality can provide more assistance for students to understand program execution and enhance the learning experience.

Author Contributions: Conceptualization, P.-W.L. and S.-H.Y.; methodology, C.-H.L.; software, P.-W.L. and S.-H.Y.; validation, C.-H.L. resources, P.-W.L. and S.-H.Y.; data curation, P.-W.L. and C.-H.L.; writing—original draft preparation, P.-W.L., S.-H.Y. and C.-H.L.; writing—review and editing, P.-W.L. and C.-H.L.; visualization, P.-W.L.; supervision, C.-H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Science and Technology Council under Grant (113-2813-C-033-012-H) and Chung Yuan Christian University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to school privacy policies.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lai, C.H. Design Interactive Response System based on Hot Option for Improving the Participation of Students: A Case Study. *Educ. J.* **2023**, *51*, 77–99.
2. Yu, C.C.; Wu, L.Y. Instructing with Cognitive Apprenticeship Programming Learning System (CAPLS) for novice computer science college freshmen. *Educ. Technol. Soc.* **2024**, *27*, 183–196.

3. Wang, J.Y.; Wang, W.Y.; Liang, J.C. Designing an Educational Chatbot for Enhanced Learning in Programming Courses. In Proceedings of the 2024 12th International Conference on Information and Education Technology (ICIET), Yamaguchi, Japan, 18–20 March 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 237–242.
4. Liang, J.-C.; Wang, J.-Y.; Chang, C.-C. The CodingHere Platform for Programming Courses. *Inf. Eng. Express* **2022**, *8*, 1–14. [CrossRef]
5. Kui, X.; Liu, W.; Xia, J.; Du, H. Research on the improvement of python language programming course teaching methods based on visualization. In Proceedings of the 2017 12th International Conference on Computer Science and Education (ICCSE), Houston, TX, USA, 22–25 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 639–644.
6. Han, G.; Kagawa, K. Towards a Web-based program visualization system using Web3D. In Proceedings of the 2012 International Conference on Information Technology Based Higher Education and Training (ITHET), Istanbul, Turkey, 21–23 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1–4.
7. Georgas, J.C.; Palmer, J.D.; McCormick, M.J. Supporting software architecture learning using runtime visualization. In Proceedings of the 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), Dallas, TX, USA, 5–6 April 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 101–110.
8. Al-Fedaghi, S.; Alrashed, A. Visualization of execution of programming statements. In Proceedings of the 2014 11th International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 7–9 April 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 363–370.
9. Suryana, N.; Salam, S.; Azmi, M.S.; Noersasongko, E. 3De-synergetic program visualization: A visual learning-aid tool for novice students. In Proceedings of the International Conference on e-Education, Entertainment and e-Management, Bali, Indonesia, 27–29 December 2011; IEEE: Piscataway, NJ, USA, 2012; pp. 133–137.
10. Simoňák, S. Algorithm visualizations as a way of increasing the quality in computer science education. In Proceedings of the 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 21–23 January 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 153–157.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.