IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol.006, LIIR396
DOI: https://doi.org/10.52731/liir.v006.396

Preliminary Practices for Beginner's Programming Course Utilizing Generative AI and Online Education

Miyuki Murata [*], Naoko Kato [†],
Tetsuro Kakeshita [‡]

Abstract

Generative AI technology is rapidly advancing and is increasingly being used to automate various processes in software development, from planning to testing. In light of these technological innovations, programming education at universities and institutes of technology must be restructured to align with software development processes using generative AI. Programming and generative AI are highly compatible, and generative AI can support a wide range of tasks, including automatic code generation, refactoring, code suggestion, answering programming-related questions, and test code generation. In this paper, we propose a beginner-level online programming course designed to utilize generative AI as a support system for programming education. We developed educational content and implemented it in a preliminary trial with a small group of university students. Learning logs and questionnaire responses were analyzed to evaluate the effectiveness of the course. Our results indicate a high level of student satisfaction with both the course content and the use of generative AI. Additionally, students demonstrated increased awareness of the importance of verifying AI-generated output and crafting appropriate prompts. These findings suggest that the integration of generative AI and on-demand learning has strong potential to enhance programming education in higher education institutions.

Keywords: Generative AI, Programming education, GitHub Copilot, Pair programming, Online education

1 Introduction

Generative AI is rapidly gaining popularity and is being used in a variety of fields—not only for text generation, but also for the generation of images, audio, 3D models, and multimodal content that integrates these modalities. Considering this, an international initiative called the "*Hiroshima AI Process*" [1] was launched by the G7 in 2023, serving as a guideline that promotes transparency, accountability, and risk management in the use of generative AI, and influencing AI policy across various countries.

By 2023, various guidelines for the use of generative AI had been published in multiple countries. "*Generative Artificial Intelligence (AI) in Education*" [2] was published in the United Kingdom

[*] National Institute of Technology, Kumamoto College, Yatsushiro, Japan
[†] National Institute of Technology, Ariake College, Ohmura, Japan
[‡] Saga University, Saga, Japan

in March. In May, "*Artificial Intelligence and the Future of Teaching and Learning*" [3] was published in the United States, followed by Japan's "*Provisional Guidelines on the Use of Generative AI in Primary and Secondary Education*" [4] in July. In September, UNESCO released its first global guideline, "*Guidance for Generative AI in Education and Research"* [5], which promotes the responsible use of generative AI tools and provides guidance on their impact on education, risks, and challenges, as well as policy directions. Given that generative AI is expected to continue evolving and becoming widely adopted, it is essential to intentionally foster both an understanding of its underlying principles and the ability to utilize it effectively.

In software development, automation using generative AI is advancing across various processes, from design to testing, and the knowledge and skills required of software engineers are expected to evolve accordingly. Conventional programming and software engineering education provided at universities and institutes of technology assumes that humans must do programming. However, we argue that a paradigm shift in programming education is necessary since humans can now program using generative AI such as GitHub Copilot [6].

The purpose of our research is to develop methodologies and tools for effective programming education utilizing generative AI and to scientifically evaluate their effectiveness to provide new value to programming education in the future, where generative AI is widely adopted. In our previous work [7], we proposed the overall structure of the educational support system and designed a set of instructional units for programming education that utilizes generative AI at the beginner and intermediate levels, respectively. In this paper, we report on the development of lesson content and the results of a preliminary trial conducted with a small group of beginner-level students.

In Section 2, we review related research. Section 3 describes the overall programming education system that utilizes generative AI, followed by an overview of the beginner-level programming course reported in this paper. Section 4 describes a method of preliminary trial. Section 5 describes the analysis based on learning logs and assignment scores. Section 6 discusses the results of the participants' questionnaire. Section 7 concludes with a summary and directions for future work.

2 Related Research

Since ChatGPT [8], the pioneer of generative AI, was released for public use in November 2022, there have been few studies on the use of generative AI in the field of education. Regarding programming education, numerous studies [9][10][11][12] have examined the impact of programming lessons incorporating generative AI on student learning. These studies have shown that generative AI is effective in helping students master programming concepts and providing individualized feedback. On the other hand, they point out the difficulty for students to identify errors made by the AI. Lepp et al. [13] investigated the impact of generative AI use in university classes and found that students who used it more frequently tended to have lower academic performance.

Several studies have explored instructional approaches to programming education. Pahi et al. [14] proposed a novel active learning approach that integrates human teaching assistants (TAs) and generative AI, and confirmed through experiments that feedback leveraging the strengths of both improved learning outcomes and student satisfaction. He et al. [15] integrated AI-assisted

adaptive learning into flipped classrooms and demonstrated not only enhanced learning outcomes but also the effectiveness of immediate and personalized feedback.

Studies have also examined the use of generative AI to support programming exercises. Kazemitabaar et al. [16] reported that a support system using OpenAI Codex for beginners improved both program completion rates and performance scores. Ueno et al. [17] reported that a personalized learning system using Generative AI can support the enhancement of learning motivation through AI-generated advice. Additionally, Leinonen et al. [18] confirmed that generative AI provides useful information for resolving programming errors, such as helping students better understand error messages.

While all the above studies assume human programmers, our study explores pair programming between a generative AI and a human programmer.

3 Programming Education utilizing Generative AI

3.1 Changes in Human Roles with the Introduction of Generative AI

Program development involves a series of steps, including requirements specification, program design, coding, debugging, and maintenance. In traditional programming, although tools have existed to support each of these steps, humans have primarily been responsible for the entire process—from defining requirements to maintenance. With the use of generative AI, each step can potentially be made more efficient and automated. However, since generative AI poses risks such as hallucinations, copyright issues, and information security concerns, it is hazardous to use its outputs as-is. Therefore, human evaluation of the AI-generated content is essential. As a result, the human role in programming shifts to issuing prompts to generative AI and verifying its outputs. In other words, humans and generative AI work collaboratively in an interactive and alternating manner throughout the development process.

Figure 1 illustrates the coding workflow using generative AI. The human provides instructions, and the generative AI performs the coding. The human then reviews the generated code to ensure it aligns with the specifications. The human may also request explanations from the generative AI to verify the validity of the code. If any issues are identified, the human provides additional instructions to revise the code. This process is repeated iteratively.

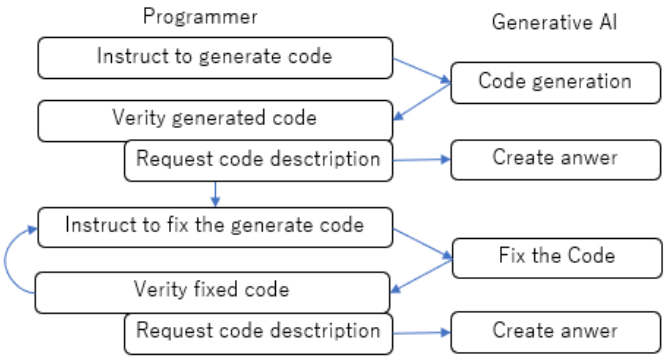

Figure 1: Coding Workflow with Generative AI

3.2 Reforms Needed in Programming Education

In programming that utilizes generative AI, coding itself is the role of AI; however, basic programming knowledge remains essential for humans to evaluate the generated output. Furthermore, programming with generative AI is expected to shift the focus of learning toward understanding fundamental concepts and algorithms through the code and explanations proposed by AI, as well as applying these algorithms in practical contexts. To have AI generate the desired code, appropriate prompts are necessary. Therefore, it becomes increasingly important to educate students in problem-solving methods—such as correctly understanding the problem and being able to decompose it into smaller subproblems—so that they can effectively instruct AI.

When utilizing generative AI, it is essential to understand its underlying mechanisms and associated issues to make informed decisions that consider both ethics and the responsibilities that come with its adoption. It is also worth noting that there is a growing need to acquire proficiency in code-generating AI tools and to learn new programming methods that incorporate these tools.

3.3 Overview of the Beginner-Level Programming Course

In beginner-level programming education, it is essential to emphasize fundamental programming skills and problem-solving abilities, while also teaching the basics of debugging and code review. Additionally, providing opportunities to engage with new tools and technologies helps to stimulate students' interest and enables them to adapt to modern programming environments.

Table 1: Learning Objectives for Beginner-Level Subjects

Introduction to Programming	
Objective	Students will be able to understand the basic concepts of programming and create simple programs.
Content	• Data types and variables, operations • Basic control statements (if statement, for statement, while statement) • Basic input and output operations (Console, File) • Definition and use of functions • Basic data structures (list, dictionary) • Utilizing libraries • Testing and debugging
Algorithms and Data Structures	
Objective	Students will be able to understand basic algorithms and data structures and implement them.
Content	• Sorting algorithms (Bubble sort, Insertion sort, Selection sort, etc.) • Search algorithms (Linear search, Binary search, etc.) • Basic data structures (Stack, Queue, Tree, Marge sort, etc.) • Big O notation (Order notation) • Dynamic search problem and data structure • Graph definition and search
Problem-Solving Through Programming	
Objective	Students will learn problem-solving methods and develop the ability to solve concrete problems through programming.
Content	• Problem decomposition and algorithm design • Pseudocode and flowcharts • Hands-on practice through simple projects

Taking these considerations into account, we designed three subjects listed in Table 1. Python was chosen as the programming language for these subjects. Python is widely used in fields, such as data science and AI, and has already been adopted in beginner-level programming education at Saga University, which is the target of this study. For these reasons, it is well suited for introductory programming education.

4 Preliminary Trial of Beginner-Level Programming Education

This section describes the content developed for each subject in the preliminary trial of beginner-level programming education, along with the corresponding lesson plan. The developed content was provided to students using Moodle.

4.1 Developed Content

4.1.1 Introduction to Programming

In the "Introduction to Programming", Google Colaboratory (Colab) was adopted as the practical environment. Colab is free to use and runs on a web browser, which reduces the burden of environment setup for learners. Additionally, the generative AI tool Gemini can be used within Colab.

As content, we created instructional pages and videos explaining how to use Colab, how to use Gemini, and how to proceed with the exercises. For the main textbook, we adopted "*Introduction to Python Programming*" [19]. This textbook comprehensively covers all the instructional topics for the subject, as well as additional topics. It is also available in a notebook format, allowing learners to open the files in Colab and study by checking the sample code and exercises directly within the environment.

4.1.2 Algorithms and Data Structures

In this subject, students learn the fundamental concepts of algorithms and data structures. The learning content encompasses all the algorithms and data structures required to complete the assignments in the subject "Problem-Solving Through Programming."

As content, we prepared lecture videos, slides, and quizzes for each learning topic. Learners first watch the lecture videos for each topic and then take the corresponding quizzes. They are encouraged to use generative AI to support their understanding of the lecture content and to review explanations of questions they answered incorrectly on the quizzes.

The lecture videos and quizzes were created based on selected materials from the "Algorithms and Data Structures", which was previously used at Saga University. The quizzes include 2 to 4 questions for each learning topic. Since there is no limit on the number of quiz attempts, learners can retake the quizzes as needed to deepen their understanding. The quizzes are presented in bullet-point format using natural language. Students are asked to fill in missing parts of an algorithm or to determine the state of variables given specific input data. The quizzes assess not only the final state of variables but also their intermediate states throughout the execution process.

4.1.3 Problem-Solving Through Programming

In this subject, students engage in programming exercises to solve problems based on the content learned in "Introduction to Programming" and "Algorithms and Data Structures". Python, the same programming language used in the introductory subject, was adopted. Visual Studio Code (VS Code) was selected as the programming environment, and GitHub Copilot was used as the generative AI tool.

GitHub Copilot is trained on open-source code from GitHub, making it more suitable for programming assistance compared to Gemini, which is trained on general web data. Another objective of using a different environment was to enable comparison with Colab and Gemini, which were used in the introductory subject. At the time of the experiment, students could use GitHub Copilot free of charge by registering with GitHub Education.

The content of this subject is divided into three parts: (1) explanations of program design and coding, (2) instructions for preparing the development environment, and (3) the programming assignments. For Part (1), lecture materials and videos from the "Software Engineering" subject, which had already been conducted at Saga University, were selected. For Part 2, we created videos explaining how to install VS Code and how to install and use GitHub Copilot. Additionally, we introduced several introductory videos about GitHub Copilot available on YouTube. For Part (3), we produced a video explaining the steps for completing the programming assignments.

The assignment problems were selected from previous edition of "Pasocon Koshien (PCK) ," a programming competition for high school students hosted by the University of Aizu. Each problem in the contest is assigned a score ranging from 2 to 20 points according to its difficulty, and this scoring system is considered a reliable indicator of problem difficulty. Since this subject is designed for beginner-level learners, we selected 10 problems with a maximum score of 6 points. An example of one of the problems is shown in Figure 2.

Number Similar to 2023
2023 can be expressed as the product of three prime numbers: 17×17×7. Among only two of these numbers are the same (17). For a given positive integer N, if it can be expressed as $N=p\times p\times q$ using two different primes p and q, we define N as a **number similar to 2023**.

Task
Given a positive integer N write a program to determine whether N is a number similar to 2023.

Input
The input consists of a single positive integer N in the following format:

N

Where A single line contains a positive integer N(1<=N<=1,000,000,000).

Output
If N is a number similar to 2023, output **"Yes"** on a single line. Otherwise, output **"No"** on a single line.

Figure 2: Assignment Example

Learners upload the source code of their programs, developed on their own PCs, to Moodle. The submission deadline for all assignments was set to one week after the final class session. Submissions received after the deadline were also accepted, and no points were deducted for late submissions.

For assignment evaluation, we used the Aizu Online Judge (AOJ) [20], an online judging system operated by the University of Aizu, which hosts the PCK programming contest. The evaluation

rubric is shown in Table 2. In some cases, program submissions were marked as incorrect due to formatting errors in the input. However, since the focus of this subject is to assess students' understanding of algorithms and data structures for problem-solving, input/output formatting issues were not penalized. For such submissions, only the input/output sections were appropriately corrected by the instructor, and then the solutions were submitted to AOJ for judgment.

Table 2: Grading Rubric for Assignments

Condition	Score
Correct output is obtained for all test cases.	100
Correct output is obtained for some test cases.	90
Correct output is obtained for all sample cases listed in the problem statement.	80
The program runs but does not meet the assignment requirements.	60
The program contains errors and does not run.	50

4.2 Lesson Plan

As a preliminary trial, we conducted on-demand classes for three subjects with students from Saga University. The course content was provided via Moodle, and a separate course page was prepared for each subject. The participating students were five to seven fourth-year students majoring in computer science. The implementation schedule is shown in Table 3.

The course ran from October 11 to November 28, 2024, with each Friday designated as class time. Students individually worked on the content during these scheduled sessions. Any unfinished activities or assignments were completed outside of class hours.

Table 3: Trial Course Schedule

Subject	Start Date	End Date	Assignment Deadline	# of Students
Introduction to Programming	10/11/2024 (Fri.)	10/24/2024 (Thu.)	No Assignment	7
Algorithms and Data Structures	10/25/2024 (Fri.)	11/7/2024 (Thu.)	11/8/2024 (Fri.)	6
Problem-Solving Through Programming	11/8/2024 (Fri.)	11/28/2024 (Thu.)	11/29/2024 (Fri.)	5

5 Analysis of Learning Data

5.1 Learning Logs

We analyzed the access logs collected by Moodle. Figure 3 shows the number of accesses to each course. Here, "Number of accesses" refers to the total number of times content within each course was accessed.

"Introduction to Programming" had the fewest accesses, while "Algorithms and Data Structures" had the most. Since both courses included either quizzes or external learning content outside of Moodle, it is assumed that students engaged with these components on a regular basis. There was no notable variation in access frequency among individual students.

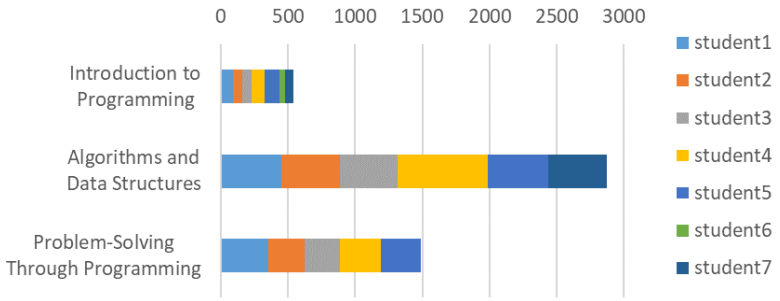

Figure 3: Number of Course Accesses in Moodle

Figure 4 illustrates the variation in the number of accesses to each course over a 28-day period, commencing from the trial's initiation. The horizontal axis represents the number of days since the course began, with Day 1 corresponding to the course start date.

For all subjects, the number of accesses was higher on class days. In "Algorithms and Data Structures" and "Problem-Solving Through Programming," there were also frequent accesses on days other than the scheduled class days. These two subjects included quizzes and assignments, suggesting that students worked on them outside of class hours. Additionally, in "Problem Solving Through Programming," there was a noticeable increase in access just before the assignment deadline.

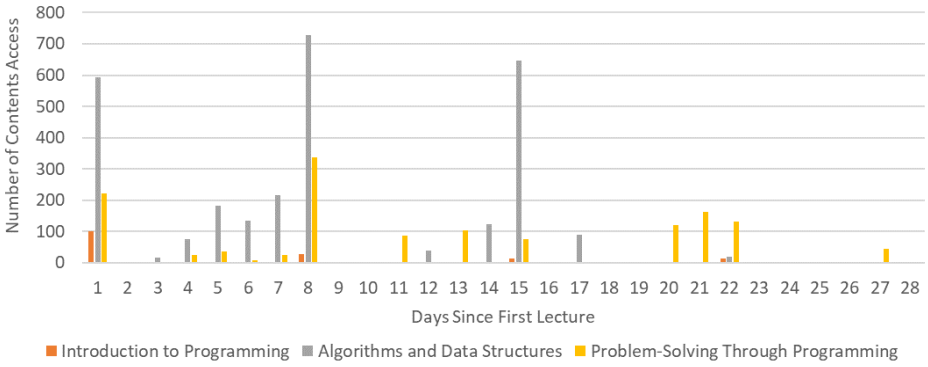

Figure 4: Transition of Course Accesses in Moodle Over Time

The content can be broadly classified into three categories: (1) materials in HTML or PDF format, (2) explanatory videos for the materials, and (3) quizzes or assignments. Table 4 shows the number of accesses for each content category.

In "Introduction to Programming", the number of accesses dropped to zero after the third week. Since the course duration was set to two weeks, it is assumed that students completed their learning within the intended period. Although the course duration for "Algorithms and Data Structures" was also set to two weeks, the deadline for the quizzes was scheduled for the end of the third week, and thus access was observed during that time.

In "Problem-Solving Through Programming," access to the assignments increased from the second week onward. This was likely because the programming environment had to be set up in the

first week, enabling students to begin working on the assignments from the second week. The course duration was three weeks, and the assignment deadline was set to four weeks after the start, which explains the continued access until that point.

Table 4: Number of Course Accesses grouped by Content type

Course	Introduction to Programming		Algorithms and Data Structures			Problem-Solving Through Programming		
Content-Type	Docu-ment	Video	Docu-ment	Video	Assign-ment	Docu-ment	Video	Assign-ment
Week 1	144	85	84	49	941	85	88	49
Week 2	29	2	29	36	735	50	57	311
Week 3	0	0	20	18	562	10	11	286
Week 4	0	0	0	0	2	1	5	135

5.2 Evaluation of Quizzes and Assignments

5.2.1 Algorithms and Data Structures

Each quiz in this subject is scored on a 10-points scales. There was no limit on the number of quiz attempts. The final average score was 9.2, with 4 out of 6 students achieving a perfect score. Given that the average score on the first attempt was 7.6, it can be inferred that students retook the quizzes multiple times to achieve full marks. Figure 5 shows the average score and number of attempts on the first attempt for each quiz.

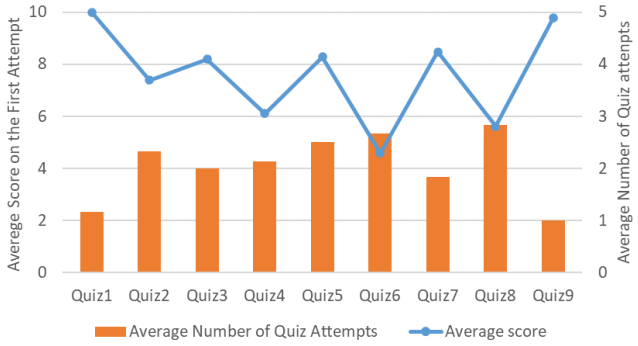

Figure5: Average Score on the First Attempt and Overall Average Number of Attempts per Quiz

According to Figure 5, Quizzes 1 and 9 had high average scores and were attempted only once on average. Quiz 1 covered introductory topics in "Algorithms and Data Structures", and the questions involved selecting the correct terminology. Quiz 9 included problems that required determining the results of graph traversal using techniques such as breadth-first search. As graph traversal problems are widely known, the students likely found them easy to understand and solve.

Quizzes 6 and 8, which had low average scores, are considered here. These quizzes also had higher average numbers of attempts. Quiz 6 included two questions related to dynamic search problems. Question 1 required students to complete a dynamic hashing algorithm using separate chaining. In the first attempt, three out of six students answered this question partially incorrectly. Question 2 asked students to determine the state of a red-black tree after inserting a given value. In the first attempt, all six students answered incorrectly, and five of them left several blanks unanswered. Although similar examples were presented in the lecture videos, the results suggest

that the students' understanding of red-black trees was insufficient.

Quiz 8 consisted of four questions on advanced sorting algorithms such as merge sort and quick sort. In each question, students were required to determine the state of the data at each step as the algorithm progressed. In this format, a single incorrect answer often leads to subsequent errors in later steps. On the other hand, Quiz 7 also included four questions of the same format, but students achieved higher scores on their first attempt and needed fewer attempts overall. Since Quiz 7 focused on basic sorting algorithms, this suggests that advanced sorting algorithms like merge sort and quick sort are more challenging for students to understand.

5.2.2 Problem-Solving Through Programming

Figure 6 shows the distribution of problem point values and the average scores of students. A weak negative correlation was observed between the point values and average scores, with a correlation coefficient of -0.28. However, the p-value was 0.45, which is substantially greater than the commonly used significance level of 0.05. Therefore, this result is not considered statistically significant.

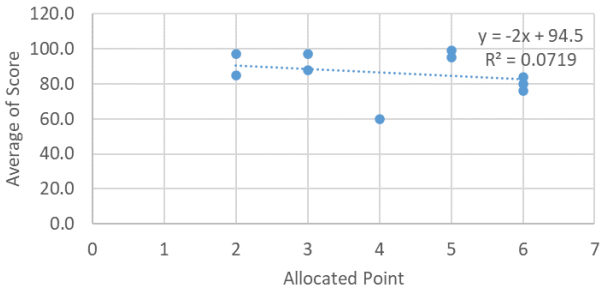

Figure 6: Distribution of Problem Point Values and Average Scores

The grading results are shown in Table 5. The problem with the highest average score was Problem 6, with a score of 99.0. Problems with an accuracy rate below 80.0 were Problems 5, 8, and 10.

Table 5: Problem points assigned by the PCK contest and average student scores

Problem	1	2	3	4	5	6	7	8	9	10
Problem point	2	2	3	3	4	5	5	6	6	6
Average score	85.0	97.0	97.0	88.0	**60.0**	**99.0**	95.0	**76.0**	84.0	**80.0**

For submissions scoring below 60 points, although the expected output was provided in the problem description along with the input, students were unable to produce correct results. This suggests that they failed to reflect the intended algorithm in their implementation. Submissions scoring between 80 and 90 points were able to produce correct output for some inputs but likely failed to handle edge cases or exceptions appropriately.

In the following, we examine the results for the low-performing problems—Problems 5, 8, and 10. GitHub Copilot was used to assist in analyzing the errors in the submitted programs.

In Problem 5, students were asked to determine the number of groups formed by people arranged in a circle, based on information about whether they were holding hands. In many of the submitted programs that used depth-first search to count the number of groups, the logic failed to

account for cases where the last person and the first person were holding hands, or where all individuals were connected. This suggests that the analysis and understanding of the problem were insufficient. Students who scored higher on this problem wrote programs based on the observation that the number of unconnected individuals corresponds to the number of groups. Instead of checking each person sequentially, they devised a more efficient algorithm, which led to correct results.

Problem 8 required students to determine the degree to which pasta was cooked, based on past data that related cooking time to doneness. Incorrect answers generally failed to express the correct conditions in the branching logic, indicating a lack of alignment with the intended interpretation of the problem.

Problem 10 was a variation of the ball rearrangement problem, in which there were restrictions on how balls could be swapped. The task was to determine whether a given arrangement could be sorted in ascending order. One student who scored 50 points failed to implement the required conditions correctly. Other students who attempted to simulate the sorting process exceeded the time limit. In contrast, students who scored higher applied a modified version of the bubble sort algorithm that performed only a single pass to check whether sorting was possible, enabling them to complete the task within the time constraint.

6 Student Questionnaire

6.1 Questionnaire Overview

After the course was completed, a questionnaire was administered for the participants, as shown in Table 6. The number of respondents was six for both “Introduction to Programming” and “Algorithms and Data Structures”, and five for “Problem Solving Through Programming”.

Table 6: Questionnaire Items for Course Participants

No.	Question	Type
1.	How satisfied were you with the class format using on-demand videos?	5-pt scale
2.	Please explain your reason. Also, if you have any suggestions or ideas for improvement, please share them.	Open-ended
3.	How many total hours did you spend watching course videos and working on assignments?	Fixed-choice
4.	How clear were the explanations and materials in the video content?	5-pt scale
5.	How helpful were the exercises in helping you understand the experimental content?	5-pt scale
6.	Please select all generative AI services you used to work on exercises.	MCQ
7.	How satisfied were you with the generative AI services?	5-pt scale
8.	Please explain your reason. If you did not use generative AI, please also explain why.	Open-ended

Note: MCQ = Multiple-Choice Question; Fixed Choice = Predefined answer options.

6.2 Satisfaction with the Course and Content

Figures 7 through 9 show the results of the questionnaire regarding the on-demand class format and course content. For all questions, most students responded with either “Very satisfied” or

"Satisfied". In response to Question 2, students across all subjects commonly appreciated the ability to study at their own pace and to review the materials multiple times.

In "Algorithms and Data Structures," one student noted that taking quizzes after watching the videos helped deepen their understanding. However, another student commented that although the video explanations were clear, it became difficult to follow how the data changed when the problems grew more complex. In "Problem-Solving Through Programming," a student reported difficulty in applying for GitHub Education.

Figure 10 illustrates the time required to complete each course. In "Problem-Solving Through Programming," one student spent more than 30 hours, but the others completed the course within 20 hours. Based on these results, we suggest that integrating the three subjects developed in this study and adjusting the volume of assignments appropriately, it is feasible to structure them as a one-credit course for universities or technical colleges.

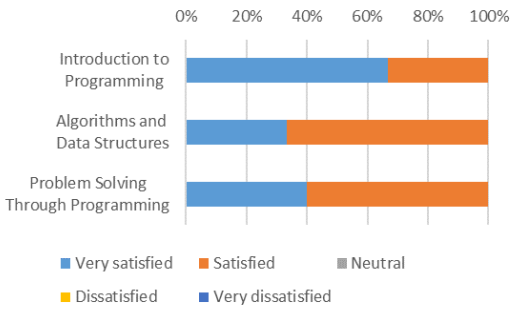

Figure 7: Satisfaction with the on-demand format

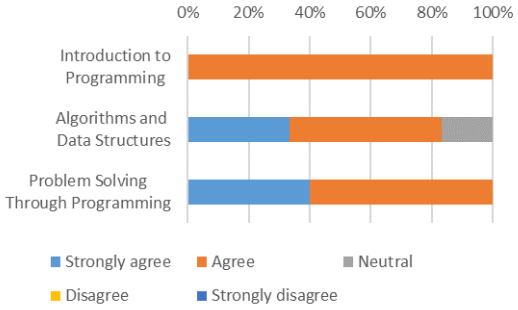

Figure 8: Clarity of the documents and videos

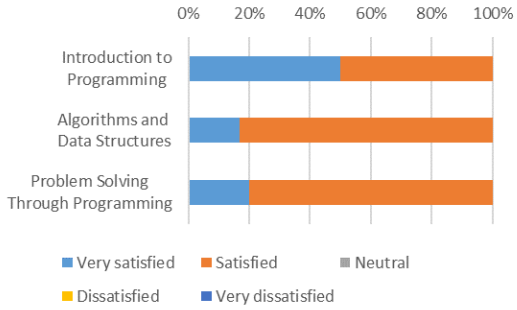

Figure 9: Usefulness of the exercises

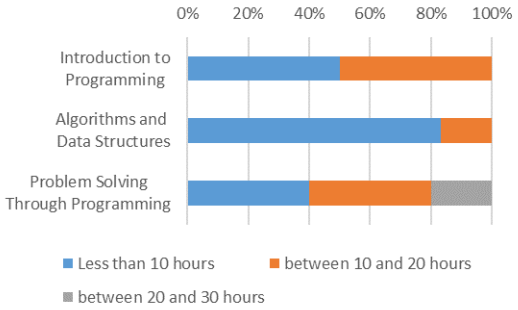

Figure 10: Total hours spent on the course

6.3 Use of Generative AI

Figure 11 shows the results of the questionnaire regarding the use of generative AI. Although some students responded "Neutral" in "Introduction to Programming" and "Algorithms and Data Structures", the overall responses indicate high level of satisfaction.

In the free-text responses, two students in "Introduction to Programming" expressed dissatisfaction with the generative AI's responses being in English or the Japanese translations being inaccurate. In "Algorithms and Data Structures", three students noted frequent errors in the AI's output or an inability to obtain the desired results.

On the other hand, positive responses were observed in all subjects, with students stating that the explanations provided by the generative AI helped them understand difficult concepts and that the code suggestions were useful. In particular, for "Problem Solving Through Programming", students commented on the usefulness of the AI's code suggestions and autocomplete functionality. However, there were also responses indicating that the AI-generated code did not produce the correct output. Some students emphasized the importance of critically evaluating AI's output. This evaluation process is essential when programming with generative AI.

Additionally, one student mentioned that it took considerable time to design appropriate algorithms and data structures to generate effective prompts. This suggests that the ability to construct suitable data structures and algorithms remains important, even when using generative AI for coding.

Figure 12 illustrates the generative AI tools that students utilized during the courses. Multiple responses were allowed. Note that ChatGPT and Google Gemini were used in their free versions.

In "Algorithms and Data Structures," one student reported not using any generative AI, stating in the free-response section that they believed it was essential to understand the material independently. Another student who indicated that they used GitHub Copilot referred to coding in their comment; however, since there were no coding tasks in this subject, it is possible that they were confusing it with "Problem-Solving Through Programming."

In "Problem-Solving Through Programming," where VS Code and GitHub Copilot were used as the programming environment, all students reported using GitHub Copilot. According to the free responses, some students also utilized ChatGPT when considering algorithms and data structures. These responses suggest that students chose different generative AI tools depending on their specific learning objectives.

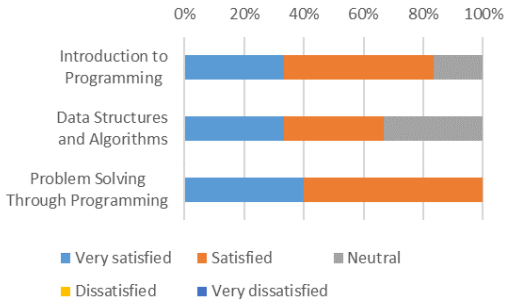

Figure 11: Satisfaction with generative AI

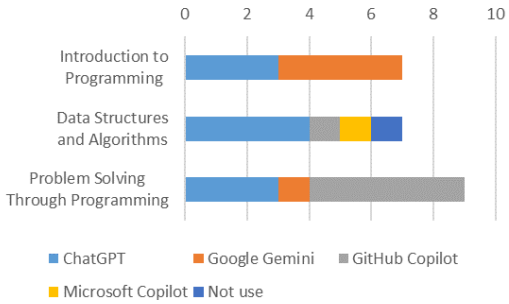

Figure12: Generative AI Used

7 Conclusion and Future Work

In this paper, we propose a beginner-level curriculum for programming education that incorporates the use of generative AI and develops corresponding on-demand educational content. As a preliminary trial, the content was implemented with a small group of students, and its effectiveness was evaluated through analysis of learning logs and student questionnaires. The results indicated a high level of satisfaction with both the content and the use of generative AI. Moreover,

it was found that students became aware of the importance of verifying AI-generated output and crafting appropriate prompts.

Based on these outcomes, we plan to expand the target group in fiscal year 2025 to include second- and third-year students in the Department of Intelligent Information Systems at Saga University, offering the courses in an on-demand format. In preparation for this, we intend to refine and improve the instructional content of "Algorithms and Data Structures" and the assignments in "Problem-Solving Through Programming."

In the current implementation, students were required to apply for GitHub Education to use GitHub Copilot at no cost. As a result, some students experienced delays in setting up the programming environment and starting the assignments. However, on December 18, 2024, GitHub Copilot Free [21] was released, which allows the use of generative AI without requiring a GitHub Education account, albeit with some usage limitations. This development is expected to simplify environment setup and promote the use of generative AI among beginners. The performance of generative AI is rapidly improving, and in some cases, it is even surpassing human capabilities. For this reason, the importance of programming education based on the premise of using generative AI, which is the aim of this research, is expected to increase even further in the future.

Acknowledgment

We appreciate the students of Faculty of Science and Engineering, Saga University, for their cooperation in this practice. This research is supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant Number 24K06431.

References

[1] G7, "Hiroshima AI Process," Oct. 2023; https://www.soumu.go.jp/hiroshimaaiprocess/en/index.html.

[2] The Department for Education of the United Kingdom, "Generative artificial intelligence (AI) in education," Mar. 2023; https://www.gov.uk/government/publications/generative-artificial-intelligence-in-education.

[3] The United States Department of Education, "Artificial intelligence and the future of teaching and learning," May. 2023; https://tech.ed.gov/ai-future-of-teaching-and-learning/.

[4] The Ministry of Education, Culture, Sports, Science and Technology of Japan, "Provisional guidelines on the use of generative AI in primary and secondary education" (in Japanese), Jul. 2023; https://www.mext.go.jp/content/20230710-mxt_shuukyo02-000030823_003.pdf.

[5] UNESCO, "Guidance for generative AI in education and research," Sep. 2023: https://www.unesco.org/en/articles/guidance-generative-ai-education-and-research.

[6] GitHub Inc., "GitHub Copilot"; https://docs.github.com/ja/copilot.

[7] Anonymous, "Research Vision for Paradigm Shift and Support Tools in Programming Education Utilizing Generative AI," Proceedings of the Symposium on Information

Education(SSS2024), Vol. 2024, pp.95-102 (in Japanese).

[8] OpenAI, “ChatGPT”; https://chatgpt.com/.

[9] P. Haindl, and G. Weinberger, “Students’ Experiences of Using ChatGPT in an Undergraduate Programming Course”, IEEE Access, Vol. 12, 2024, pp.43519-43529.

[10] M. M. Rahman, and Y. Watanabe, “ChatGPT for education and research, Opportunities, threats, and strategies,” Applied Sciences, vol. 13, no. 9, 2023, p. 5783.

[11] M. Daun, and J. Brings, “How ChatGPT will change software engineering education,” in Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1(ITiCSE 2023), 2023, pp. 110-116.

[12] M. Hu, T. Assadi and H. Mahroeian, “Explicitly Introducing ChatGPT into First-year Programming Practice: Challenges and Impact,” IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), 2023; doi: 10.1109/TALE56641.2023.10398297.

[13] M. Lepp, J. Kaimre, "Does generative AI help in learning programming: Students’ perceptions, reported use and relation to performance," Computers in Human Behavior Reports, volume 18, 2025, article 100183: doi:https://doi.org/10.1016/j.chbr.2025.100642.

[14] K. Pahi, S. Hawlader, et al., "Enhancing active learning through collaboration between human teachers and generative AI", Computers and Education Open, vol. 6, 2024, article 100183; doi: https://doi.org/10.1016/j.caeo.2024.100183.

[15] Y. He, Y. Song, and Y. Ji, “Integrating AI-based adaptive learning into the flipped classroom model to enhance engagement and learning outcomes,” Education and Information Technologies, vol. 29, 2024, pp. 1845–1869.

[16] M. Kazemitabaar, J. Chow et al, “Studying the effect of AI code generators on supporting novice learners in introductory programming”, in Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems(CHI '23), no. 455, 2023, pp. 1-23.

[17] H. Ueno, Y. Tanaka, et al., “Development of a Learning Advising System Using Generative AI”, IEEE Conference Proceedings of IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), 2024, pp.693-694.

[18] J. Leinonen, A. Hellas, S. Sarsa et al., “Using Large Language Models to Enhance Programming Error Messages,” Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2023, New York, NY, USA, Association for Computing Machinery, 2023, pp. 563–569; doi: 10.1145/3545945.3569770.

[19] Mathematics and Informatics Center, “The University of Tokyo: Introduction to Python Programming,”; https://utokyo-ipp.github.io/.

[20] University of Aizu: “AIZU ONLINE JUDGE,”; https://onlinejudge.u-aizu.ac.jp/

[21] TechCrunch, “GitHub launches a free version of its Copilot,” 2024;

https://techcrunch.com/2024/12/18/github-launches-a-free-version-of-its-copilot/.