EMANATE - EDUCATIONAL
SESSIONS HIGHLIGHTS
EESH

https://www.emanate.education/

DOI: 10.70020/eesh.2024.12.5

ICEEPSY 2024
15th International Conference on Education & Educational Psychology

PROTOTYPE ADVISORY SYSTEM FOR LEARNING C PROGRAMMING USING GENERATIVE AI

Akiyoshi Wakatani (a)*, Toshiyuki Maeda (b)
*Corresponding author

(a) Faculty of Intelligence and Informatics, Konan University in Kobe, Japan, wakatani@konan-u.ac.jp
(b) Faculty of Information Sciences, Hannan University in Osaka, Japan, maechan@hannan-u.ac.jp

Abstract

The technology of generative AI using LLM (Large Language Models) has made remarkable progress, and attempts to apply the technology of program code generation to programming education have attracted much attention. In this paper, we evaluate the suitability of a virtual TA (Teaching Assistant) system for beginner-level learners using OpenAI's API, which takes C programs with errors and error messages as input to the LLM and outputs appropriate advice. Among the programming errors, the proposed system was generally effective in generating appropriate advice for syntactic and semantic errors, which is sufficient for learners to solve the problems on their own. For logical errors, the proposed system was generally effective in generating appropriate advice, although in some cases the advice was only a general explanation. Moreover, there are cases in which the prompts give appropriate advice to review the part of the sentence that includes division, so there is a possibility to obtain appropriate advice by asking for advice more than once.

Keywords: Large language model, programming error, teaching assistant, programming language, debugging tool

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference

1. Introduction

In recent years, the technology of generative AI based on LLMs (large language models) has made remarkable progress, and new systems have been presented one after another. Interactive and generative AIs such as OpenAI's ChatGPT (OpenAI, 2024) and Google's Gemini (Gemini, 2024) can generate text and natural dialogue with a certain degree of accuracy and have succeeded in attracting over 100 million users, but there are still issues such as the problem of hallucination, which causes wrong answers, and ChatGPT is trained with information only until 2021. However, the technology for code generation of programs is highly developed, and OpenAI's Code Interpreter based on GPT-4 can not only generate a Python program from the sentence "Create a python program for regression analysis using scikit learn from temperature and sales files stored in csv," but can also execute the program if appropriate data are given (OpenAI, 2024). In other words, it can be said that the level of perfection of the generated AI has reached a fairly high level with regard to the functions of creating and interpreting programs using programming languages.

On the other hand, there is a serious shortage of human resources in IT and AI, and the training of engineers who meet the demands of the world is an urgent issue. The acquisition of programming skills is one of the indispensable items in the training of IT and AI engineers. In learning programming, it is more important for students to learn how to modify their own programs than for programs to be generated automatically. In this case, it is known that the advice from a teaching assistant (TA) is effective in helping beginners learn programming, and that simply presenting a correct program does not lead to learning. For example, in the literature (Hellas et al., 2023), it is stated that providing immediate model answers is problematic from a pedagogical point of view, and it is suggested to fine-tune the LLM to not give the correct answer, but instead help the student reach the answer, leading to the conclusion that the LLM has a lot of experience with codes, but not much understanding of good pedagogy.

Attempts to improve the efficiency of programming learning using AI have already been initiated. In the literature (Kazemitabaar et al., 2023) examined the effectiveness of the OpenAI Codex in helping novice programmers create programs, and found a 1.15-fold improvement in program completion rate and a 1.8-fold improvement in performance. In addition, not only the programming ability but also the performance of the post-programming test has been improved. Moreover, Leinonen et al. in the literature (Leinonen et al., 2023) showed that the GPT-3 explanation for error messages in the Python language is useful for understanding the error content, and that the adjustment of the temperature parameter is becoming important. However, since the Python language under the study is executed by an interpreter and the error messages can be generated with both interpretation and runtime information of the program, the explanations output by the system are also considered to be accurate. Therefore, for compiler languages such as C, which has different types of compilation errors and runtime errors, the results of this study alone do not determine its usefulness.

Therefore, in this paper, we propose a prototype of a programming learning support system (a virtual TA (Teaching Assistant) system that only gives advice to programs that contain programming errors without presenting corrected code. Furthermore, we will evaluate the adequacy of the system's advice for typical error messages in the C programming language, which contains both compile errors and runtime errors.

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference

2. Programming Errors

In C programming, bugs, which are errors in the program, can be classified into three main categories: syntax errors (including lexical errors), semantic errors, and logical errors (Aho et al., 2006). The first two are called compile errors because they are often found at compile time, and the third is also called runtime errors because they are often found at runtime

Syntax errors are caused by grammatical mistakes, such as confusion between int and integer, confusion between elseif and elif and other keywords, forgetting semicolons and commas, using wrong operators, using wrong types of parentheses, and other grammar and lexical errors. These errors are caused by mis-remembered grammar and phrases. For beginners, since they are not familiar with the programming language itself, they imitate the mathematical formatting and make mistakes, which are the cause of frequent errors.

The second factor, semantic error, occurs when the syntax, the sequence of characters or phrases, is correct but the program semantics is not. For example, declaring a variable xx as both int and double is syntactically possible, but semantically incorrect since one variable cannot have two types. Also, if a function func is defined with one variable of type int as its argument, the function func cannot be called with two arguments, but no syntactic error occurs. Other examples include the use of unpermitted keyword combinations such as unsigned float in C, branching to undefined labels, pointer references by variables that are not pointer variables, and various other factors.

The third factor, logical error, is different from the syntactic and semantic errors mentioned above, and is strictly an error that can be detected only by running the program. For example, for an array x of size 10, if i=10; and then x[i+3] is used to access the array elements, a memory address error occurs. Also, in the case of the program “i=10; while(i>1)i++;”, the variable i grows infinitely large from 10, resulting in an infinite loop or overflow of the variable value. In addition to these errors, there are cases in which it is not possible to predict with certainty that an error will occur just by looking at the program, and these errors are called logical errors.

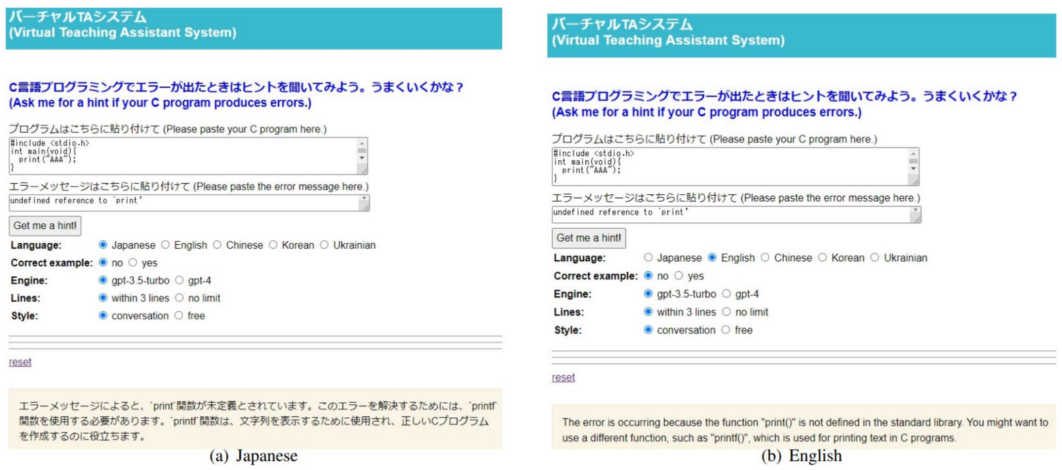
As described above, various errors occur in the programming learning process for novice programmers, and these errors cannot be solved by the novice programmers themselves. Therefore, it is expected that the TA’s advice will help beginners to easily understand the errors and improve their learning efficiency to master the programming language.

3. Virtual TA System

As mentioned in the previous sections, it is important for novice programmers to have a TA who can give them appropriate advice when they make mistakes in their own programs. Therefore, we propose a prototype of a virtual TA system (hereinafter referred to as VTA system) that displays the advice of such a TA using the OpenAI API. Figure 1 shows a screen capture of the system. The left side is the output screen in Japanese, and the right side is the output screen in English.

The current VTA system is a system only for C language, and there is an area where you can enter the original program and the error messages when you compiled and executed the program, and when you click the button “Get me a hint”, appropriate advice will be displayed.

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference



バーチャルTAシステム
(Virtual Teaching Assistant System)

C言語プログラミングでエラーが出たときはヒントを聞いてみよう。うまくいくかな？
(Ask me for a hint if your C program produces errors.)

プログラムはこちらに貼り付けて (Please paste your C program here.)

```
#include <stdio.h>
int main(void){
  print("AAA");
}
```

エラーメッセージはこちらに貼り付けて (Please paste the error message here.)

```
undefined reference to `print'
```

Get me a hint!

Language:	[✓ Japanese] English Chinese Korean Ukrainian
Correct example:	[✓ no] yes
Engine:	[✓ gpt-3.5-turbo] gpt-4
Lines:	[✓ within 3 lines] no limit
Style:	[✓ conversation] free

reset

エラーメッセージによると、`print`関数が未定義とされています。このエラーを解決するためには、`printf`関数を使用する必要があります。`printf`関数は、文字列を表示するために使用され、正しいCプログラムを作成するのに役立ちます。

(a) Japanese

バーチャルTAシステム
(Virtual Teaching Assistant System)

C言語プログラミングでエラーが出たときはヒントを聞いてみよう。うまくいくかな？
(Ask me for a hint if your C program produces errors.)

プログラムはこちらに貼り付けて (Please paste your C program here.)

```
#include <stdio.h>
int main(void){
  print("AAA");
}
```

エラーメッセージはこちらに貼り付けて (Please paste the error message here.)

```
undefined reference to `print'
```

Get me a hint!

Language:	Japanese [✓ English] Chinese Korean Ukrainian
Correct example:	[✓ no] yes
Engine:	[✓ gpt-3.5-turbo] gpt-4
Lines:	[✓ within 3 lines] no limit
Style:	[✓ conversation] free

reset

The error is occurring because the function "print()" is not defined in the standard library. You might want to use a different function, such as "printf()", which is used for printing text in C programs.

(b) English

Figure 1. Virtual TA system

The LLM (Large Language Model) used is either GPT4 or GPT-3.5-Turbo, which can be selected, and the output language can be selected (Japanese, English, Chinese, Korean, Ukrainian). In addition, it is possible to show an example of how to write a correct program in the advice (Correct example), but this function is not selected by default because it is considered better not to show it for the benefit of learning programming. Furthermore, since it is better to keep the length of the advice short, two patterns of advice length (Lines) are provided: one is 2-3 lines, and the other is no restriction on the length.

This system is a PHP program using JQuery 3.5.1 and OpenAI API, running Apache/2.4.41 web server with PHP 7.4.3 on a computer running Ubuntu 20.04 LTS OS. Figure 2 shows the OpenAI API program in PHP using CURL. The LLM used can be changed to GPT-3.5-Turbo as well as GPT4, and the appropriate prompt string is set to $message3 in the program.

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "https://
    api.openai.com/v1/chat/completions")
    ;
...
$data = array('model' => 'gpt-4');
...
$data["messages"] = array();
$message3='appropriate prompt'
$data["messages"][] = array('role' => '
    user', 'content' => $message3);
curl_setopt($ch, CURLOPT_POSTFIELDS,
    json_encode($data));
$result = curl_exec($ch);
$response = json_decode($result, true);
curl_close($ch);
```

Figure 2. OpenAI API program in PHP using CURL (some parts omitted)

The OpenAI API charges a fee based on the amount of data exchanged. The amount of data is counted in units of tokens, and it is known that one word in English is generally one token, while one character in Japanese consumes several tokens. In order to reduce the amount of data, prompts are sent in English, and the output language is set by indicating the output language in the prompt, such as “in Japanese”.

Figure 3 shows the prompt to obtain the output shown on the left in Figure 1. The prompt is phrased according to the options set by the browser, but the “in conversation style” indicates that the responses

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference

should be output in spoken form. The reason for using this option is that it is intended to be used more like a real TA, since it is assumed that in the future responses will be output not only in written form but also in spoken form.

```
My C program is as follows:
#include <stdio.h>
int main(void){
print("AAA");
}
I got the following error message.
undefined reference to ‘print’
Show me a hint to get a correct program
    in Japanese, in conversation sentence
     style, within 3 lines, but, do not
    show the correct C program.
```

Figure 3. Example of prompt

4. Evaluation

4.1. Evaluation methods

As already mentioned, we have implemented a web application that provides advice on programming errors using the OpenAI API. We provide the system with a C program containing syntactic and semantic errors, which are compilation errors, and logical errors, which are runtime errors, and their error messages, and evaluate whether the output advice is valid or not. The error messages used in the evaluation are those output by the GCC compiler (9.4.0) on a computer running the Ubuntu 20.04 LTS operating system

The C language programs were prepared in five patterns for each error, and the advice was obtained three times with the same settings, and each pattern was evaluated. The options are GPT-3.5-Turbo for the LLM (At the time this experiment was conducted, GPT-4 was not yet available.), Free for the number of lines of output (no specification of 2-3 lines), no output of the modified C program as the correct answer, and the temperature option is used as default.

The evaluation of the advice output for each error is given in the following subsections: Tab. 1, Tab. 2, and Tab. 3. The evaluation is based on a subjective evaluation by the authors. The evaluation shall be made on a 5-point scale as follows: S: The explanation is apt using line numbers or variable names, etc., A: Line numbers or variable names are not provided, but they are aptly described, B: Although aptly described, a modified program is also presented, C: Correct explanation, but only in general terms, D: It is a wrong explanation.

4.2. Syntactic errors

As shown in Table 1, five program pieces were employed, including the following syntax errors: 1. missing semicolon, 2. illegal operator, 3. missing parenthesis, 4. wrong keyword, 5. missing operator. 1. missing semicolon is a program that forgot to write the semicolon at the end of the executable statement. 2. illegal operator is a program that contains the incorrect operator usage of i=j++k;. 3. missing parenthesis is a program that forgot to write the closing parenthesis in the block of if statements. 4. wrong keyword is

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference

a program that declares variables with integer instead of int. 5. Missing operator is a program that contains assignment statements that forget to write some operators, such as i=(j+k)(j-k);.

Table 1. Evaluating advice for syntactic errors

Errors	Factors	1	2	3
A. Syntactic errors	1. missing semicolon	S	S	A
	2. illegal operator	S	S	S
	3. missing parenthesis	B	S	S
	4. wrong keyword	S	S	S
	5. missing operator	S	S	S

Figure 4 is one of the examples of 1. missing semicolon execution. It outputs a proper advice including the line numbers of the program. In addition to the example shown in Execution Example 1, error messages for syntax errors often include line numbers, so the advice often includes line numbers and variable names, etc., and generally appropriate advice (S) is output. However, there was a case in which the following advice was output for 1. missing semicolon, which did not specify the location: "Tip: Modify the arithmetic expression by adding a semicolon." Since sentence generation by LLM is non-deterministic, it is considered necessary to supplement inadequate advice with a function such as "listening back" in case of inadequate advice.

(program)

```
#include <stdio.h>
int main(void){
  int i,j,k;
  i=j+k
}
```

(error message)

```
5:8: error: expected ';' before
 '}' token
    5 | i=j+k
      |      ^
      |      ;
    6 | }
      | ~
```

(comment) Hey, it seems like your assignment of values to 'i' is missing the semicolon at the end which is causing the error. Try adding a semicolon after 'i=j+k'. Also, make sure that variables 'j' and 'k' have been initialized before they are used in operations.

Figure 4. Execution example 1 (missing semicolon)

4.3. Semantic errors

Table 2. Evaluating advice for semantic errors

Errors	Factors	1	2	3
B. Semantic errors	1. no type declaration	S	S	S
	2. type declaration error	S	S	S
	3. attribute error	S	S	S
	4. function error 1	S	S	S
	5. function error 2	B	B	S

As shown in Table 2, five program pieces were employed, including the following semantic errors: 1. no type declaration, 2. type declaration error, 3. attribute error, 4. function error 1, 5. function error 2 1. no type declaration is a program that forgot to write the declarations of some variables in the program. 2. type declaration error is a program in which some variables in the program are declared with multiple types.

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference

3. attribute error is a program that contains the disallowed keyword combination unsigned float. 4. function error 1 is a program that calls a function with one argument with two arguments. 5. function error 2 is a program that calls a function with no return value with i=func(k);.

Figure 5 shows one of the examples of execution of 1. No type declaration, and it is able to output appropriate advice after clarifying the names of variables that have not been declared.

(program)

```
#include <stdio.h>
int main(void){
  int i,j;
  i=j+k;
}
```

(error message)

```
5:7: error: ‘ ’k undeclared (first
    use in this function)
    5 | i=j+k;
      | ^
5:7: note: each undeclared
    identifier is reported only once
     for each function it appears in
```

(comment) You seem to have used a variable ’k’ that doesn’t have any value assigned to it, neither has it been declared in your program before use. A defined value is crucial for more accurate results. Therefore, before using ’k’ in your calculations, you should declare and assign a value to ’k’.

Figure 5. Execution Example 2 (no type declaration)

In addition to the example shown in Execution Example 2, error messages for semantic errors often include line numbers, so the advice often includes line numbers and variable names, etc., and generally appropriate advice (S) is output. However, in this experiment, the modified programs were displayed in several patterns even though the settings were set to deny the display of the modified programs. Although the necessary instructions are clearly indicated in the prompts, it is considered necessary to devise prompts that emphasize the instructions in order to perfect the settings.

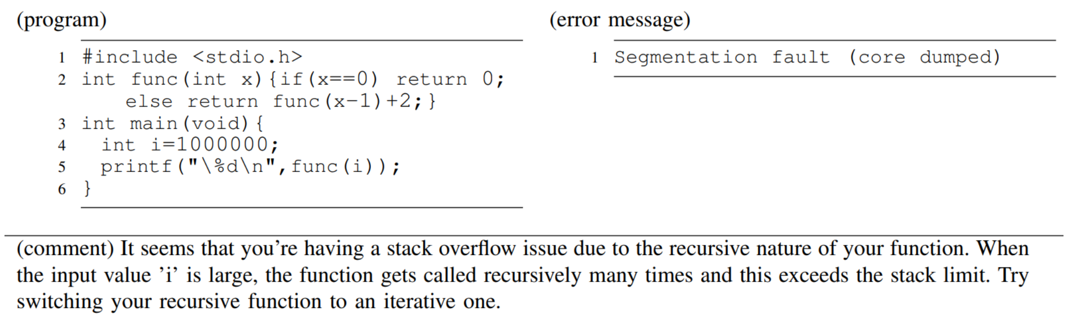
4.4. Logical errors

Table 3. Evaluating advice for logical errors

Errors	Factors	1	2	3
C. Logical errors	1. out of region access 1	S	S	S
	2. out of region access 2	S	S	S
	3. zero division 1	C	A	D
	4. zero division 2	C	S	A
	5. deep recursive call	B	S	S

As shown in Table 3, five program pieces were employed, including the following logical errors: 1. out of region access 1, 2. out of region access 2, 3. zero division 1, 4. zero division 2, 5. deep recursive call. 1. out of region access 1 is a program that references array elements with indices that exceed the declared array size. 2. out of region access 2 is also a program that refers to array elements with indices that exceed the declared array size, but the calculation of indices is rather complicated, and the only error message obtained from 1 and 2 is “Segmentation fault”. 3. zero division 1 is a program that uses the value of i/(i-10) with a variable i of value 10. 4. zero division 2 is a program that uses both i/(i-10) and i/(i+10) with a variable i of value 10, and the only error message obtained from both 3 and 4 is “floating point exception. 5. deep recursive call is a program in which the call depth of a recursive function call is so deep that the stack space is exhausted, and the only error message obtained is “Segmentation fault”.

Figure 6 is one of the execution examples of 5. deep recursive call, and it is able to output appropriate advice including the solution method after understanding that the function func is a recursive function and the number of its recursive calls is large


(program)

```
#include <stdio.h>
int func(int x){if(x==0) return 0;
    else return func(x-1)+2;}
int main(void){
  int i=1000000;
  printf("\%d\n",func(i));
}
```

(error message)

```
Segmentation fault (core dumped)
```

(comment) It seems that you're having a stack overflow issue due to the recursive nature of your function. When the input value 'i' is large, the function gets called recursively many times and this exceeds the stack limit. Try switching your recursive function to an iterative one.

Figure 6. Execution Example 3 (deep recursive call)

However, while the advice on syntactic and semantic errors is generally adequate, the advice on some logical errors is not sufficient. In the case of out of region errors, it is generally accurate (S). For example, in the case of a Segmentation fault error message, the following advice is generated to understand the content properly and to check it with respect to the array size: "The size of arrays d and c is 1000, and the array index starts at 0. Therefore, the range of indices is 0 to 999. When accessing array elements, make sure that the indices fall within these ranges". Although the learner does not know which line of the program to look at, the learner understands that the point of interest is the index of the array, as mentioned in the advice. Also, as mentioned above, the learner is able to generate appropriate advice for the deep recursive call.

On the other hand, the advice for zero division is sometimes only a general explanation (C) or incorrect (D), so there are cases where appropriate advice is not generated. For example, the advice of error (D) in 4-3 is as follows, and although it focuses on the denominator of division, the explanation is mathematically and logically incomprehensible as follows: "To correctly understand the cause of the problem, we are dividing by the value 10 of the variable i. The value of i must always be 10 since 10 is subtracted from the value of i as the denominator of the division. However, your code initializes the value of i with 10".

In addition, for 4-1, only a general explanation is given (C), and in fact, as shown below, it is difficult for the learner to know which part of the program to focus on. This error message indicates that the program has raised a floating-point exception as follows: "Tip: A floating point exception is being raised because the program is dividing by zero. Pay attention to the number to be divided".

On the other hand, there are cases in which the prompts give appropriate advice to review the part of the sentence that includes division, so there is a possibility to obtain appropriate advice by asking for advice more than once. However, this is a heavy burden for the learner, and we will consider measures such as improving the prompts in the future.

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference

5. Conclusions

A virtual TA system that takes C programs with errors and error messages as input to LLM and outputs appropriate advice was prototyped for beginner-level learners using OpenAI's API, and its adequacy was evaluated.

Among the programming errors, the VTA system generates generally adequate advice for syntactic and semantic errors, and the advice is sufficient for learners to solve the problems on their own. However, in compiler languages such as C, compile-time information, such as the correspondence between line numbers in the program and instruction numbers of machine language instructions, or between variables in the program and their addresses in memory, is not maintained at runtime, making it difficult to generate error messages using information in the program in case of runtime errors. Therefore, for logical errors, which are one of the runtime errors, in some cases, only general explanations were given, but in other cases, the system was able to generate appropriate advice. Thus, in general, the effectiveness of the proposed system was confirmed. Also, adding a "ask for additional advice" function in case of inadequate advice is expected to improve the usefulness of the system.

In the future, we would like to contribute to software education by increasing the number of program examples including program errors and evaluation items to make the evaluation more precise, and by confirming the effectiveness of this system by applying it to other languages, such as the Java language.

Data Availability Statement

Data is available upon request.

Declaration of Conflicts Interests

The authors would like to declare that they have no conflict of interest to disclose.

Ethical Statement

Our institution does not require ethics approval for reporting individual cases or case series.

References

Aho, A. V., Lam, M. S., Sethi, S. R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools* (2nd Ed.). Pearson Education.

Gemini. (2024). *Gemini*. https://gemini.google.com/app

Hellas, A., Leinonen, J., Sarsa, S., Koutcheme, C., Kujanpää, L., & Sorva, J. (2023). Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. *Proceedings of the 2023 ACM Conference on International Computing Education Research* (Vol. 1, pp. 93-105). https://doi.org/10.1145/3568813.3600139

Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, (pp. 1-23). https://doi.org/10.1145/3544548.3580919

Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., & Becker, B. A. (2023). Using Large Language Models to Enhance Programming Error Messages. *Proceedings of the 54th ACM*

https://doi.org/10.70020/eesh.2024.12.5
Corresponding Author: Akiyoshi Wakatani
Selection and peer-review under responsibility of the Organizing Committee of the conference

Technical Symposium on Computer Science Education (Vol. 1, pp. 563-569). https://doi.org/10.1145/3545945.3569770

OpenAI. (2024). *ChatGPT*. https://chat.openai.com/