Olympiads in Informatics, 2026, Vol. 20, 43–60


DOI: https://doi.org/10.15388/ioi.2026.04

# Research on the Effectiveness of GPT-4o and GPT-5.4 Models in Solving Olympiad Programming Tasks on the Eolymp Platform

Andrii MYKHALKO[1], Oleksandr MITSA[2], Yurii HOROSHKO[3], Hanna TSYBKO[4], Viktor SHAKOTKO[5]

[1]Faculty of Information Technologies, Uzhhorod National University. Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava.

[2]Department of Information Management Systems and Technologies, Uzhhorod National University.

[3,4]Department of Computer Science and Computer Engineering, Taras Shevchenko National University "Chernihiv Colehium".

[5]Department of Technological and Professional Education, Oleksandr Dovzhenko Hlukhiv National Pedagogical University.

e-mail: [1]likespro.eth@gmail.com, [2]alex.mitsa@gmail.com, [3]horoshko_y@ukr.net, [4]a.tsb@ukr.net, [5]vv0304@gmail.com

**Abstract**. The object of the research is the process of solving Olympiad programming tasks with Ukrainian statements from the Eolymp platform using the large language models GPT-4o and GPT-5.4.

As a result we discovered that LLM average efficiency in solving sports programming tasks with Ukrainian statements was nearly identical for C++ and Python, averaging around 32% for GPT-4o and 56% for GPT-5.4. It was revealed that LLMs effectiveness when providing Ukrainian and English statements for the same tasks was approximately the same ($p>0.05$).

It was found that the proportion of tasks completely solved by LLMs was about 13% for GPT-4o and 37 % for GPT-5.4 of the total number considered in this study. The result for GPT-4o is comparable with the results of solving similar tasks with English statements on the Codeforces platform, while the result for GPT-5.4 is significantly larger, however, the number of tokens used by the new model GPT-5.4 is almost 3 times larger.

**Keywords**: Olympiad programming, large language model, GPT-4o, GPT-5.4, Eolymp, Python, C++.

## 1. Introduction

In the past few years, artificial intelligence (AI) has radically changed the situation in many areas, from learning (Albahijan *et al.,* 2025; Yashina, 2024) and prediction (Kotsovsky & Batyuk, 2024; Kotsovsky, 2025) to solving medical problems (Mykhalko *et al.,* 2024; Mykhalko *et al.,* 2023) and classification (Mitsa *et al.*, 2025; Kotsovsky, 2024). In many cases, it performs as well as humans, or even better, and this has already raised concerns about the possible replacement of ordinary workers by AI (AI study, 2023). One such area where some companies are already laying off people in favor of Large Language Models (LLMs) is programming. So far, however, AI tools are not "smart" enough to fully delegate most tasks in this area. Nevertheless, in the hands of an experienced programmer, LLMs can be a great helper (Dakhel *et al.*, 2023; Ferdiana, 2024).

As neural networks are currently developing at an extremely high rate (Geche *et al.,* 2022), there exist many studies that measure the effectiveness of new models in programming subfields using various metrics and methods. For example, it has been found that OpenAI's LLM GPT-4 is extremely successful in solving problems from the LeetCode platform (Coello *et al.*, 2023) and from the Most Basic Python Problems (MBPP) set from Google researchers (Yang *et al.,* 2025). Also, in the field of intelligent code completion, it has been studied that while older AI tools mainly rely on previously known patterns, newer ones can generalize and synthesize existing knowledge into new, more complex constructions of completed code (Hou & Ji, 2024). However, the question of the effectiveness of the GPT-5.4 model in this aspect remains unexplored.

## 2. Analysis of literary data and problem statement

In (Jain *et al.,* 2025), the problem of comparing the performance of large language models (LLMs) with human programmers in solving algorithmic problems was considered. The authors set he goal to find out to what extent modern LLMs (GPT-4, GPT-3.5, Gemini, Claude, Llama, etc.) are able to compete with participants in online programming platforms, as well as to determine the impact of different hinting strategies and programming languages on the quality of the results obtained.

The research (Jain *et al.,* 2025) was based on experiments in the LeetCode and GeeksforGeeks environments, where the tests covered tasks of varying complexity levels. Several interaction strategies with the models were tested: from one-time queries to iterative schemes with code execution and verification. The solutions generated by LLM were compared with the results of thousands of participants of these platforms.

The results showed that GPT-4 significantly outperformed other models and in many cases reached or even exceeded the level of most human programmers. In particular, on both platforms, GPT-4 outperformed more than 85% of participants on average and was in the top 10% in about 70% of cases. The use of iterative schemes provided the largest performance gains. In addition, GPT-4 demonstrated the ability to transform solutions between different programming languages (Python, Java, C++, JavaScript), while maintaining or even improving their correctness. The performance of the generated programs in terms of

speed and memory consumption was comparable to human solutions.

It should be noted that the study (Jain *et al.,* 2025) has limitations: the tasks used do not fully reflect the complexity of Olympiad and professional programming; the effectiveness of LLM largely depends on the correct formulation of prompts; tasks that require extensive design or deep algorithmic thinking were not considered.

In (Görmez *et al.,* 2024), the task was to investigate the development of ChatGPT models in programming, comparing their performance in solving algorithmic problems of varying complexity. Particular attention was paid to the efficiency in terms of runtime and memory usage, as well as how the choice of programming language (Python, Java, C++) affects the results.

There were selected 15 tasks from LeetCode, which were classified by complexity. The GPT-3.5, GPT-4, GPT-4o models were tested. Each task was solved in three languages: Python, Java, C++ . The code generated by the models was executed 10 times for each task, and the execution time and memory consumption were measured. Then, statistical analysis was performed, namely, two-way ANOVA and Tukey's HSD test to compare the results between models and languages. The following results were obtained.

The programming language has a significant impact on execution time and memory usage: C++ performed best among the three languages; Python was the slowest and most resource-intensive. Java is of middle level. As for comparing the GPT-3.5, GPT-4, and GPT-4o models, no statistically significant differences in performance were found for most tasks. The models cope successfully with the tasks of medium and low complexity, but performance dropped significantly at high complexity. Although ChatGPT models show great potential in helping with algorithmic tasks, their practical utility is limited in cases where resources (time, memory) or high level of complexity are important. The choice of programming language is important: using C++ can significantly improve performance if resource constraints matter.

Therefore, the authors emphasize that ChatGPT is better viewed not as a replacement for a human programmer, but as a complementary tool, especially for tasks of medium complexity, refactoring, or for getting help with code.

In (Coello *et al.,* 2024) the following questions were considered:

1. What LLMs are used in software engineering tasks - their architecture, properties, areas of application.
2. How are data / datasets used for such models collected and processed - quality, purity, data preparation.
3. What are the optimization strategies and methods for evaluating the effectiveness of models in software engineering tasks.
4. For which specific tasks did the studied models give the best or noticeable results; where have they been successfully applied.

The studied models include both general-purpose large language models (LLMs) (e.g., GPT-like, Codex, CodeGen), and models specialized in code.

Regarding datasets, it is pointed out that many studies use large publicly available sets of code and text (GitHub, public repositories, other datasets). But there are problems with data quality: noise, duplicates, mixing of test and training data (data leakage), heterogeneity, lack of detailed annotation.

The study found that LLMs perform well in code generation, code completion/autocompletion, code summarization/documentation, and error detection/fixing.

However, there are tasks where the results are less unambiguous — tasks with large contexts, specific domains, when it is necessary to understand the behavior of the entire software project, architectural decisions, performance, security, etc.

In (Semerikov *et al.*, 2025), the goal was to evaluate the effectiveness of ChatGPT models (versions GPT-3.5 and GPT-4) in programming on example of tasks in Python, comparing them with other popular large language models, namely Google Bard, Anthropic Claude, and Bing Chat.

The Mostly Basic Python Problems (MBPP) problem set was used, a subset of problems (approximately 460) verified by Google. All models received only a statement and the name of the function that should be in the solution. The code generated by the models was tested through tests with a set of test cases. There were also stages when the models were given feedback: if the first option did not work, the test cases were returned to the model, giving the opportunity to correct the solution.

It was found out that GPT-4 showed the highest performance — ~87.5% of correct solutions; GPT-3.5 and Bing Chat — also had high results (slightly lower than GPT-4). GPT-3.5 about 83%, Bing about 81.96%. Bard and Claude had significantly lower indicators: Bard ~76.16%, Claude ~71.43%. As for the code quality, non-GPT-based models more often generated longer, less efficient code; GPT models — more concise, more efficient. When providing feedback, GPT-4 managed to fix almost all previously unsuccessful problems (14 out of 16), while Bard solved only 5.

Thus, the models considered have great potential as assistants in programming, especially for basic or intermediate-level tasks, but they cannot completely replace a human programmer - control, correction, and testing are required.

In (Souza *et al.*, 2025), the authors consider current approaches to deploying large language models (LLMs) on resource-constrained edge devices. The study is devoted to analyzing the technical, energy, and security aspects of using LLMs without constant connection to powerful cloud servers.

The authors describe:

- hybrid solution architectures that combine edge and cloud computing;
- optimization methods (quantization, pruning, knowledge distillation) that allow running GPT-like models on mobile or embedded devices;
- ducational and scientific scenarios for using LLM on edge devices, in particular in learning labs and remote environments;
- prospects for the development of edge-LLM, including issues of ethical control, local personalization, and system autonomy.

In general, the work outlines a new paradigm for AI development—the transfer of artificial intelligence directly to user devices, which opens up opportunities for safer, faster, and more independent intelligent systems.

It is worth noting separately the study of the effectiveness of various LLMs in solving Olympiad programming tasks. It was found that relatively new models that can run locally on sufficiently powerful computers completely solve up to 63.6% of tasks from the Codeforces platform (OpenAI, 2024), and proprietary ones - about 89% (Mykhalko & Mitsa, 2025).

However, most such studies analyze the power of AI on tasks with English statements. At the same time, there are almost none for tasks with Ukrainian statements (TIOBE, 2025), so this area requires further research.

## 3. Purpose and objectives of research

The purpose of our research is to investigate the effectiveness of using the large language models GPT-4o and GPT-5.4 in solving Olympiad programming tasks on the Eolymp platform in the programming languages C++ and Python with Ukrainian and English statements.

To achieve this goal, the following research tasks were formulated:

1. To investigate the effectiveness of the large language models GPT-4o and GPT-5.4 in solving Olympiad programming tasks with Ukrainian statements;
2. Compare the performance of AI when it works with Ukrainian and English statements;
3. Compare the efficiency of solving the above-mentioned tasks using the C++ and Python programming languages.

## 4. Materials and methods of research

### *4.1 Object, hypothesis of research*

The object of research is the process of solving Olympiad programming problems with Ukrainian statements from the Eolymp platform using the large language models GPT-4o and GPT-5.4.

The hypothesis of the research is that the GPT-4o and GPT-5.4 models are able to ensure the correctness of solving Olympiad programming problems in Ukrainian at a level comparable to problems in English.

### *4.2 Input data and evaluation methodology*

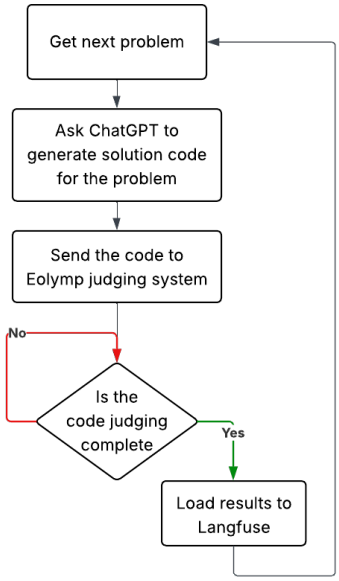


Figure 1. Block diagram of the algorithm of the developed program performance in the first mode.

The tasks were taken from the automatic system of program testing Eolymp, in the amount of 100 for each level of complexity for each of the LLMs. To automate the study, we developed a program in Python that supported two operating modes: adding new tasks and testing LLM on them. In the first mode (Figure 1), the program, using the Eolymp API, automatically found new tasks of the selected level of complexity, which are available in at least Ukrainian and English, and suggested that the operator copy the Ukrainian/English statement and enter it into the system. Then the program loaded this data into the database on the Langfuse platform and added clarifying information, in particular, the level of complexity and task ID.

In the second mode of operation (Figure 2), the developed program tested LLM in a fully automatic mode on all tasks of the selected level entered into the Langfuse database. First, it took the next task for processing, then asked LLM to write code for it using a query in Ukrainian (Я надам тобі умову задачі. Напиши програму для її розв'язку мовою програмування <мова>. Пиши тільки код, без пояснень.) and English (I will give you a problem statement. Write a program to solve it in the Python programming language. Write only the code, no explanations.), and after receiving the proposed solution from LLM, the code was sent for automatic testing by the Eolymp system via its API. In the same way, the code score and verdict were obtained. When testing problem solutions, the Eolymp system returned 2 indicators - the percentage of passed tests (solution score) and verdicts: Compilation Error (CE), Wrong Answer (WA), Time Limit Exceeded (TL), Memory Limit Exceeded (ME), Runtime error (RE). After a successful solution evaluation, the results were recorded in Langfuse and the program moved on to testing the next problem.

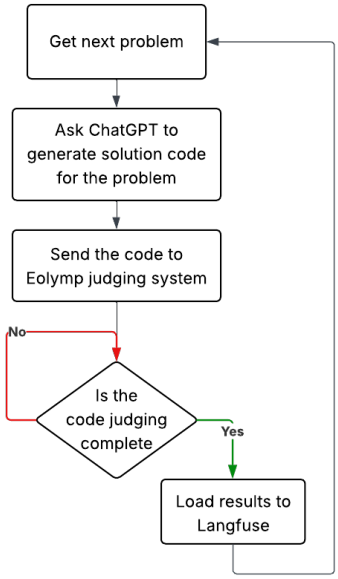


Figure 2. Block diagram of the algorithm of the developed program performance in the second mode.

After successfully testing each of the selected LLMs on all tasks using C++ and Python, the results were exported from Langfuse as JSON and converted to Microsoft Excel. These files were then used for statistical analysis of the results in TIBCO Statistica 12.

Normality of distribution of the results obtained during the experiment was checked using the Shapiro-Wilk's W test. The significance of the difference in indicators between groups was determined using the Mann-Whitney U test. When comparing proportions, the Fisher exact p, two-tailed test was used. The 95% Confidence Interval (CI) was calculated using the Wilson method. The indicators are presented as M±m. The difference was considered statistically significant at $p<0.05$.

### *4.3 Ethical considerations*

All requests have been pre-anonymized and do not contain any personal data. Use of the API complies with the terms of the OpenAI policy.

## 5. Research results

### *5.1 Research on solving Olympiad programming tasks on the Eolymp platform in the programming language C++ with Ukrainian statements*

During the research, it was found that GPT-4o generated on average 368.48±12.56 C++ code tokens per task. And the higher was the complexity, the more tokens were generated on average (341.65±18.96, 370.53±22.86, 393.27±23.08 for easy, medium, and hard tasks,

respectively). However, this difference was not statistically significant ($p > 0.05$). The average efficiency of this LLM in solving tasks was about 32%.

Regarding the GPT-5.4 model - on average, 1101,28±98,35 tokens of C++ code were generated per task. For tasks of easy, medium and hard complexity there were generated accordingly (1101,28±98,35, 1404,58±114,66, 1577,88±106,7) tokens. However, the indicated difference was not statistically significant ($p > 0.05$). The average efficiency of this LLM in solving tasks is about 56%.

The highest efficiency of GPT-4o was reached when solving tasks of easy level of complexity (Table 1). The average productivity of this LLM when solving tasks of medium and hard levels was almost 2 times lower (for both $p < 0.05$).

GPT-5.4 demonstrated almost 2 times higher efficiency then GPT-4o in solving problems of all levels (Table 2, Figure 3).

Table 1. Average efficiency (GPT-4o, Ukrainian statements)

| Level \ Language | C++ | Python |
|---|---|---|
| Easy | 45,82±4,23 % | 45,87±4,18 % |
| Medium | 26,19±3,16 % | 29,57±3,32 % |
| Hard | 23,76±3,08 % | 20,04±2,58 % |

Table 2. Average efficiency (GPT-5.4, Ukrainian statements)

| Level \ Language | C++ | Python |
|---|---|---|
| Easy | 73,79±3,95 % | 72,46±3,86 % |
| Medium | 53,26±4,16 % | 47,42±4,17 % |
| Hard | 41,50±4,27 % | 45,68±4,06 % |

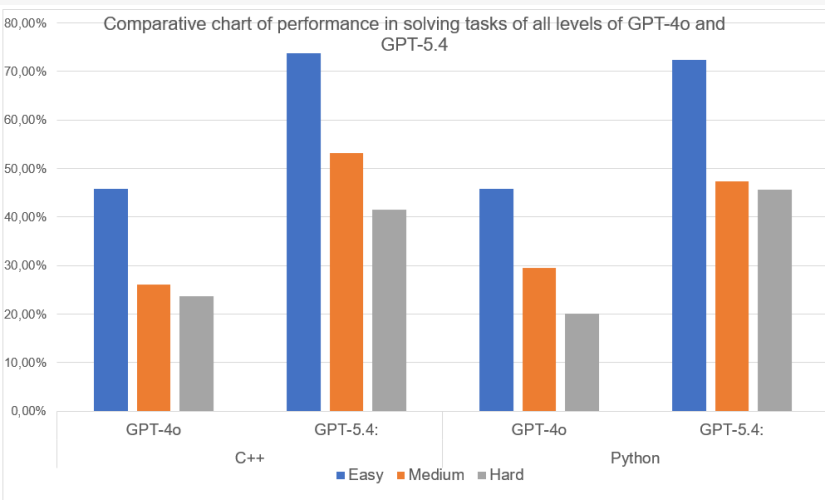


Figure 3. Comparative chart of performance in solving tasks of all levels of GPT-4o and GPT-5.4

The code that allowed to obtain a completely correct solution was generated by GPT-4o for 38 (12.67%) tasks. Most often the correct solutions were obtained for tasks of easy level of complexity, while the frequency of complete solution of tasks of medium and hard level of complexity was significantly lower (Table 3).

The GPT-5.4 model generated code to obtain a completely correct solution for 120 tasks (40%). The efficiency of solving tasks of easy level of complexity was more than 2 times higher than in the case of GPT-4o, and for tasks of medium and hard levels - 6 times higher (Table 4).

Table 3. Completely correct solutions (GPT-4o, C++, Ukrainian statements)

| Level \ Statistics | Number of tasks | % | 95% CI | p |
|---|---|---|---|---|
| Easy | 29 | 29,00 | 21,01-38,54 | $p < 0,05$ |
| Medium | 5 | 5,00 | 2,15-11,18 | $p < 0,05$ |
| Hard | 4 | 4,00 | 1,57-9,84 | $p < 0,05$ |
| Totally | 38 | 12,67 | | |

Table 4. Completely correct solutions (GPT-5.4, C++, Ukrainian statements)

| Level \ Statistics | Number of tasks | % | 95% CI | p |
|---|---|---|---|---|
| Easy | 65 | 65,00* | 55,25-73,64 | $p < 0,05$ |
| Medium | 33 | 33,00 | 24,56-42,69 | $p < 0,05$ |
| Hard | 22 | 22,00 | 15,00-31,07 | $p < 0,05$ |
| Totally | 120 | 40,00 | | |

*- the difference is statistically significant when comparing with Medium and Hard

However, there were tasks that were not completely solved, i.e. rated as 0% (GPT-4o - 101 tasks, GPT-5.4 - 60 tasks). Such cases occurred most often among tasks of medium complexity. Among tasks of hard and easy complexity, the frequency of such cases was lower (Table 5, Table 6).

Table 5. Completely wrong solutions (GPT-4o, C++, Ukrainian statements)

| Level \ Statistics | Number of tasks | % | 95% CI | p |
|---|---|---|---|---|
| Easy | 24 | 24,00 | 16,69-33,23 | $p < 0,05$ |
| Medium | 41 | 41,00 | 31,87-50,80 | $p < 0,05$ |
| Hard | 36 | 36,00 | 27,27-45,76 | $p < 0,05$ |
| Totally | 101 | 33,67 | | |

Table 6. Completely wrong solutions (GPT-5.4, C++, Ukrainian statements)

| Level \ Statistics | Number of tasks | % | 95% CI | p |
|---|---|---|---|---|
| Easy | 10 | 10,00* | 5,52-17,44 | p < 0,05 |
| Medium | 27 | 27,00 | 19,27-36,43 | p < 0,05 |
| Hard | 23 | 23,00 | 15,84-32,15 | p < 0,05 |
| Totally | 60 | 20,00 | | |

*- the difference is statistically significant when comparing with Medium and Hard

The most frequent reason for incomplete solution when using both models was the WA verdict (Table 7, Table 8). At the same time, the frequency of other verdicts was significantly lower. Moreover, with increasing complexity level of tasks, the WA verdict occurred more often, while the frequency of others was almost the same.

It is also worth noting that the ML (Memory Limit Exceeded) verdict was obtained only once for hard task when using GPT-4o.

Table 7. Reasons for incomplete solutions (GPT-4o, C++, Ukrainian statements)

| Verdicts | CE | | WA | | TL | ME | RE |
|---|---|---|---|---|---|---|---|
| Level \ Statistics | % | % | 95% CI | p | % | % | % |
| Totally | 11,45 | 72,52 | 66,82-77,57 | p < 0,05 | 10,69 | 0,38 | 4,96 |

Table 8. Reasons for incomplete solutions (GPT-5.4, C++, Ukrainian statements)

| Verdicts | CE | WA | TL | ME | RE |
|---|---|---|---|---|---|
| Level \ Statistics | % | % | % | % | % |
| Easy | 11,43 | 71,43* | 5,71 | 2,86 | 0 |
| Medium | 13,43 | 76,12* | 5,97 | 2,99 | 1,49 |
| Hard | 5,13 | 84,62* | 7,69 | 2,56 | 0 |
| Totally | 10,00 | 77,39 | 6,46 | 2,80 | 0,50 |

*- the difference is statistically significant when comparing with other verdicts

### *5.2 Research on solving Olympiad programming tasks on the Eolymp platform in the programming language Python with Ukrainian statements*

When generating code in Python using GPT-4o, there were generated on average 284.05±10.60 tokens per task, that is almost 23% less in comparison with this indicator when using C++. The average efficiency when solving tasks in this language was 31.84±2.07%. Among the reasons for incomplete tasks solutions for easy level tasks, WA verdicts occurred most often and were 1.83 times more often than in C++ ($p < 0.05$). Other verdicts, except RE, occurred almost as often. In completely incorrect solutions, WA and TL verdicts had almost the same frequency of occurrence as in the case of C++ ($p>0.05$).

When generating code in Python using GPT-5.4 here were generated on average 1220,51±85,09 tokens per task.

Data on completely correct solutions, completely wrong solutions, and reasons for incomplete solutions when using the GPT-5.4 model are given in Tables 9, 10, 11.

Table 9. Completely correct solutions (GPT-5.4, Python, Ukrainian statements)

| Level \ Statistics | Number of tasks | % | 95% CI | p |
|---|---|---|---|---|
| Easy | 58 | 58,00* | 48,21-67,20 | p < 0,05 |
| Medium | 26 | 26,00 | 18,40-35,37 | p < 0,05 |
| Hard | 22 | 22,00 | 15,00-31,07 | p < 0,05 |
| Totally | 106 | 35,33 | 30,14-40,90 | |

*- the difference is statistically significant when comparing with Medium and Hard

Table 10. Completely wrong solutions (GPT-5.4, Python, Ukrainian statements)

| Level \ Statistics | Number of tasks | % | 95% CI | p |
|---|---|---|---|---|
| Easy | 11 | 11,00* | 6,25-18,63 | p < 0,05 |
| Medium | 26 | 26,00 | 18,40-35,37 | p < 0,05 |
| Hard | 22 | 22,00 | 15,00-31,07 | p < 0,05 |
| Totally | 59 | 19,67 | 15,56-24,54 | |

*- the difference is statistically significant when comparing with Medium

Table 11. Reasons for incomplete solutions (GPT-5.4, Python, Ukrainian statements)

| Verdicts | CE | WA | TL | ME | RE |
|---|---|---|---|---|---|
| Level \ Statistics | % | % | % | % | % |
| Easy | 0 | 69,05* | 21,43 | 0 | 9,52 |
| Medium | 10,81 | 60,81* | 20,27 | 2,70 | 5,41 |
| Hard | 3,85 | 79,49* | 11,54 | 3,85 | 1,28 |
| Totally | 4,89 | 69,78 | 17,75 | 2,18 | 5,40 |

*- the difference is statistically significant when comparing with other verdicts

### *5.3 Research on solving Olympiad programming tasks on the Eolymp platform in the programming language C++ with English statements*

When GPT-4o was provided the task statements in English, the average number of tokens of the generated answer code was 375.76±13.46, which was not significantly higher compared to the cases when the statements were given in Ukrainian ($p>0.05$). At the same time, as in the case of solving tasks in Ukrainian, the increase in the complexity of the tasks was accompanied by an increase in the number of tokens of the generated solution code (333.98±18.87, 384.04±23.73, 409.25±26.32 for tasks of easy, medium and hard complex-

ity, respectively). Although the average amount of code for tasks of medium and hard complexity was slightly higher when the statements were given in English, the indicated difference was statistically insignificant ($p>0.05$).

The average performance of this LLM when solving tasks with English statements was 30.00±2.00%, which was almost the same as the similar indicator for the Ukrainian language.

When GPT-5.4 was provided the task statements in English, the average number of tokens of the generated answer code was 979,29±52,15. At the same time, as in the case of solving tasks in Ukrainian, the increase in the complexity of the tasks was accompanied by an increase in the number of tokens in the generated solution code (737,11±115,67, 924,95±96,29, 1099,47±114,61 for tasks of easy, medium and difficult complexity, respectively).

Tables 12, 13 present the average efficiency indicators of GPT-4o and GPT-5.4 when solving tasks with English statements, depending on the level of complexity, M ± m.

Table 12. Average efficiency (GPT-4o, English statements)

| Level \ Language | C++ | Python |
|---|---|---|
| Easy | 41,02±4,01 % | 41,41±4,13 % |
| Medium | 26,57±3,06 % | 24,37±2,94 % |
| Hard | 22,41±2,99 % | 17,79±2,46 % |

Table 13. Average efficiency (GPT-5.4, English statements)

| Level \ Language | C++ | Python |
|---|---|---|
| Easy | 70,01±4,16 % | 64,25±4,26 % |
| Medium | 49,49±4,18 % | 50,96±4,14 % |
| Hard | 43,97±4,13 % | 40,90±4,07 % |

The efficiency of solving tasks of different levels of complexity using GPT-4o and GPT-5.4 (Tables 12, 13) was almost the same as in the case of the Ukrainian language.

In the case of using GPT-4o, among the reasons for incomplete solutions, as in the case of Ukrainian statements, the verdict WA was most often found (73.37%, 95% CI: 68.15-78.60). Other verdicts occurred much less often (9.63%, 11.48%, 4.81% and 0.37% for CE, TL, RE and ME, respectively). In general, these indicators were similar to those when statements were provided in Ukrainian ($p>0.05$), in particular by complexity levels. Completely incorrect solutions, i.e. solutions rated at 0%, most often received the verdict WA (66.69%, 95% CI: 57.69-75.69), less often – CE (26.63%, 95% CI: 18.22-35.05). The frequency distribution of verdicts for these solutions was also the same as for tasks in Ukrainian.

When using GPT-5.4, there were 114 completely solved tasks (61 - of easy level of complexity, 29 - medium, 24 - hard). There were 54 completely unsolved tasks (rated at 0%) (13 - of easy level of complexity, 22 - medium, 19 — hard).

Data on the reasons for incomplete solutions are given in Table 14.

Table 14. Reasons for incomplete solutions (GPT-5.4, C++, English statements)

| Verdicts | CE | WA | TL | ME | RE |
|---|---|---|---|---|---|
| Level / Statistics | % | % | % | % | % |
| Easy | 2,56 | 87,19* | 7,69 | 2,56 | 0 |
| Medium | 4,23 | 85,92* | 5,63 | 2,82 | 1,41 |
| Hard | 1,32 | 90,79* | 3,95 | 3,95 | 0 |
| Totally | 2,70 | 87,97 | 5,76 | 3,11 | 0,47 |

*- the difference is statistically significant when comparing with other verdicts

### *5.4 Research on solving Olympiad programming tasks on the Eolymp platform in the programming language Python with English statements*

When generating tasks solutions with English statements in Python, LLM GPT-4o generated code with a volume of 285.66±10.78 tokens, that is 23.98% less than in C++ ($p<0.05$), and almost the same as for Python when statements were provided in Ukrainian ($p>0.05$). By complexity levels, the dynamics of this indicator were exactly the same as with Ukrainian statements (256.49±15.61, 287.10±18.67, 313.39±21.09, for all – $p>0.05$).

The efficiency of GPT-4o in solving tasks in English for Python was 27.86±1.96%, which did not have a statistically significant difference when compared with the performance of GPT-4o in English for C++ and Ukrainian for Python. When comparing the results depending on the level of complexity, it became clear that GPT-4o generates the least efficient code in Python when providing statements in English, in comparison with other configurations described in this work. However, this difference was not statistically significant ($p>0.05$).

At the same time, the number of both completely correct solutions: 32 (easy - 26, medium - 4, difficult - 2), and completely wrong solutions: 99 (easy - 24, medium - 38, difficult - 37), was almost the same as when using Python with Ukrainian statements and using C++ with English ones.

When generating task solutions with English statements in Python, LLM GPT-5.4 generated code with a volume of 800,19±55,52 tokens, that is 23.98% less than in C++ ($p<0.05$), and almost the same as for Python with Ukrainian statements ($p>0.05$). By complexity levels, the dynamics of this indicator were exactly the same as with Ukrainian statements (637,84±90,02, 831,61±98,07, 931,14±92,06, for all – $p>0.05$).

The number of completely solved tasks was 102 (where easy — 50, medium — 30, difficult — 22). There were 59 completely unsolved tasks (rated at 0%) (of which easy level — 18, medium — 19, difficult — 22). It can be seen that when switching to the new LMM,

the number of completely solved problems increased almost 3 times, while the number of completely unsolved ones decreased almost half. At the same time, the number of tokens used increased significantly.

The most common cause of incomplete solutions for both models was the WA verdict (for GPT-4o - 70,53 %, for GPT-5.4 - 78,02 %), while its frequency of occurrence was almost the same as with other pairs of "programming language/statement language".

Table 15 presents the reasons for incomplete solutions in the case of using GPT-5.4.

Table 15. Reasons for incomplete solutions (GPT-5.4, Python, English statements)

| Verdicts | CE | WA | TL | ME | RE |
|---|---|---|---|---|---|
| Level \ Statistics | % | % | % | % | % |
| Easy | 0 | 78,00* | 14,00 | 0 | 8,00 |
| Medium | 1,43 | 71,43* | 17,14 | 1,43 | 8,57 |
| Hard | 0 | 84,62* | 8,97 | 2,56 | 3,85 |
| Totally | 0,48 | 78,02 | 13,37 | 1,33 | 6,81 |

*- the difference is statistically significant when comparing with other verdicts

## 6. Conclusions

As a result of the research, it was discovered that the average efficiency of LLM GPT-4o in solving sports programming taskss in Ukrainian when using C++ and Python was almost the same and amounted to about 32%. For the GPT-5.4 model, this result is about 56%, which is almost 2 times higher than the result of GPT-4o. Such closeness of results can be explained by the fact that, on the one hand, Python is the most popular programming language today (Bucaioni *et al.*, 2024), due to which LLMs had much more code for training. On the other hand, in the field of sports programming, C++ is most often used. Thus, LLMs may have had approximately the same amount of code of both languages for training on specific tasks of such type, as on Eolymp.

During the work, it was determined that the performance of both LLMs when providing task statements in Ukrainian and English for the same tasks was approximately the same ($p>0.05$). This can be explained by the fact that they may have also trained on Ukrainian tasks, and since the tasks had Ukrainian statements, these models understand tasks of such style better in Ukrainian. At the same time, there are much more resources in English, and as a result, both models had a larger data set for training in this language. Thus, these two factors could mutually compensate each other when providing task statements in different languages.

It was found that the proportion of tasks solved completely by LLM GPT-4o, was about 13%, and GPT-5.4 - 37 %, that is almost 3 times larger.

The result obtained for GPT-4o almost coincides with the result of solving similar tasks on the Codeforces platform with English statements (Bucaioni et al., 2024), and the result for GPT-5.4 exceeds them.

However, the results obtained cannot be fully and objectively compared, since the sample from Codeforces may have a different level of task complexity and a different distribution of difficult, medium, and easy tasks. There is also a study of the effectiveness of ChatGPT in solving LeetCode problems (OpenAI, 2024), which demonstrates a several times higher percentage of complete solutions - from 75% to 100%. However, the problems on this site are of a slightly different type and are less focused on sports programming, so these results cannot be compared with ours either.

It is worth noting the practical absence of CE verdicts for solutions in Python. Since one of features of Python is that most of the code is translated directly from the program execution time, almost all errors in the code were detected by the Eolymp system already at the testing stage and, as a result, resulted in a RE verdict.

It is also important to note that the GPT-5.4 model uses almost three times as many tokens when solving tasks as GPT-4o.

It is planned to do a research on samples of tasks, in which authors can use dialect or regionally colored words (Mitsa *et al.,* 2023), in order to analyze the influence of this factor on the ability of ChatGPT system to solve tasks.

*Conflict of interest*

The authors declare that they have no conflict of interest regarding this research, including financial, personal, authorship or other, that could influence the research and its results presented in this article.

*Funding*

The research was conducted without financial support.

*Data availability*

Data will be provided upon reasonable request.

*Use of artificial intelligence tools*

The authors used artificial intelligence technologies within the permissible framework to conduct the research (the article analyzes the capabilities of artificial intelligence) and improve readability and grammar when writing the article.

## References

AI study: Over 60 per cent use Artificial Intelligence at work – almost half of all employees are worried about losing their jobs. (2023, August 28). Deloitte. https://www.deloitte.com/ch/en/about/press-room/ai-study-almost-half-of-all-employees-are-worried-about-losing-their-jobs.html.

Albahijan, N., Alsuraibi, H., Alotaibi, J. & Alotaibi, K. (2025). Artificial intelligence in education. International Journal of Computers and Informatics, 4(1), 9–61. https://doi.org/10.59992/ijci.2025.v4n1p1.

Bucaioni, A., Ekedahl, H., Helander, V. & Nguyen, P. T. (2024). Programming with ChatGPT: How far can we go? Machine Learning With Applications, 15, 100526. https://doi.org/10.1016/j.mlwa.2024.100526.

Coello, C. E. A., Alimam, M. N. & Kouatly, R. (2023). Effectiveness of ChatGPT in coding: A comparative analysis of popular large language models. Digital, 4(1), 114–125. https://doi.org/10.3390/digital4010005.

Coello, C. E. A., Alimam, M. N. & Kouatly, R. (2024). Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models. Digital 2024, 4(1), 114-125. https://doi.org/10.3390/digital4010005.

Dakhel, A. M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C. & Jiang, Z. M. (2023). GitHub Copilot AI pair programmer: Asset or Liability? Journal of Systems and Software, 203, 111734. https://doi.org/10.1016/j.jss.2023.111734.

Ferdiana, R. (2024). The Impact of Artificial Intelligence on Programmer Productivity. international conference on software engineering and information technology (icoseit) 2024. https://www.researchgate.net/publication/378962192_The_Impact_of_Artificial_Intelligence_on_Programmer_Productivity.

Geche, F., Mitsa, O., Mulesa, O. & Horvat, P. (2022, October 4–7). Synthesis of a two cascade neural network for time series forecasting. In Proceedings of the 2022 IEEE 3rd International Conference on System Analysis & Intelligent Computing (SAIC) (pp. 1-5). Kyiv, Ukraine: IEEE. https://doi.org/10.1109/SAIC57818.2022.9922991.

Görmez, M. K., Yılmaz, M. & Clarke, P. M. (2024). Large Language Models for Software Engineering: A Systematic Mapping Study. Communications in Computer and Information Science Systems, Software and Services Process Improvement, 64-79, 2024. https://doi.org/10.1007/978-3-031-71139-8_5.

Hou, W. & Ji, Z. (2024). Comparing Large Language Models and Human Programmers for Generating Programming Code. Advanced Science, № 8. https://doi.org/10.1002/advs.202412279.

Yang, R., Dai, J., Vasilakis, N. & Rinard, M. (2025, April 7). Evaluating the generalization capabilities of large language models on code reasoning. arXiv.org. https://arxiv.org/abs/2504.05518.

Yashina, I. A. (2024). Artificial intelligence in teaching programming to students of pedagogical university. Open Education, 28(4), 23–32. https://doi.org/10.21686/1818-4243-2024-4-23-32.

Jain, R., Thanvi, J. & Subasinghe, A. (2025). The evolution of ChatGPT for programming: a comparative study. Engineering Research Express, № 1, 2025. https://doi.org/10.1088/2631-8695/ada51d.

Kotsovsky, V. & Batyuk, A. (2024) Towards the design of bithreshold ANN regressor. In 19th IEEE International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2024. Lviv, October 16–19, pp. 1–4. IEEE. http://doi: 10.1109/CSIT65290.2024.10982560.

Kotsovsky, V. (2024). Learning of multi-valued multithreshold neural units. In CEUR Workshop Proceedings, volume 3688, pp. 39–49. https://ceur-ws.org/Vol-3688/paper4.pdf.

Kotsovsky, V. (2025). Multithreshold neurons with smoothed activation functions. In CEUR Workshop Proceedings, volume 3983, pp. 93–102. https://ceur-ws.org/Vol-3983/paper7.pdf.

Mykhalko, A. & Mitsa, O. (2025). Efficiency of code generation by artificial intelligence gpt-4o for ukrainian-language tasks of various complexity on the eolymp platform. Zenodo. https://doi.org/10.5281/zenodo.15032717.

Mykhalko, Y. O., Filak, Y. F., Dutkevych-Ivanska, Y. V., Sabadosh, M. V. & Rubtsova, Y. I. (2024). From open-ended to multiple-choice: evaluating diagnostic performance and consistency of ChatGPT, Google Gemini and Claude AI. Wiadomości Lekarskie, 77(10), 1852–1856. https://doi.org/10.36740/wlek/195125.

Mykhalko, Y., Kish, P., Rubtsova, Y., Kutsyn, O. & Koval, V. (2023). From text to diagnose: chatgpt's efficacy in medical decision-making. Wiadomości Lekarskie, 76(11), 2345–2350. https://doi.org/10.36740/wlek202311101.

Mitsa, O., Sharkan, V., Maksymchuk, V., Varha, S. & Shkurko, H.: Ethnocultural, Educational and Scientific Potential of the Interactive Dialects Map. In 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), pp. 226-231. IEEE (2023). http://doi.org/10.1109/SIST58284.2023.10223544.

Mitsa, O., Voloshchuk, Y., Levchuk, O. & Petsko, V. (2025). A Comparative Study of Machine Learning Algorithms and the Prompting Approach Using GPT-3.5 Turbo for Text Categorization. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham. https://doi.org/10.1007/978-3-031-84228-3_13.

OpenAI. (2024, September 12). Learning to reason with LLMs. Retrieved May 16, 2025, from https://openai.com/index/learning-to-reason-with-llms/.

Semerikov, S. O., Vakaliuk, T. A., Kanevska, O. B., Moiseienko, M. V., Donchev, I. I. & Kolhatin, A. O. (2025). LLM on the edge: the new frontier. In Proceedings of the 5th Edge Computing Workshop (Edge @ 2025), Zhytomyr, Ukraine, April 4, 2025 (CEUR Workshop Proceedings, Vol. 3943, pp. 137–161). ISSN 1613-0073.

Souza, D., Gheyi, R., Albuquerque, L., Soares, G. & Ribeiro, M. (2025, April 9). Code Generation with Small Language Models: A Deep Evaluation on Codeforces. arXiv.org. https://arxiv.org/abs/2504.07343.

TIOBE Index for October 2025. https://www.tiobe.com/tiobe-index/

**Andrii Mykhalko**
Student
Faculty of Information Technologies
Uzhhorod National University
Student
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
E-mail: likespro.eth@gmail.com
ORCID: https://orcid.org/0009-0004-5964-3008
ResearcherID: OPM-9649-2025

**Oleksandr Mitsa**
Doctor of Technical Sciences, Professor
Department of Information Management Systems and Technologies
Uzhhorod National University
Narodna sq., 3, Uzhhorod, Ukraine, 88000
E-mail: alex.mitsa@gmail.com
ORCID: https://orcid.org/0000-0002-6958-0870
ResearcherID: G-6060-2019

**Horoshko Yurii**
Doctor of Pedagogic Sciences, Professor, Head of the Department
Department of Computer Science and Computer Engineering
Taras Shevchenko National University “Chernihiv Colehium”
st. Hetman Polubotko, 53, Chernihiv, Ukraine, 14000
Email: horoshko_y@ukr.net
ORCID: 0000-0001-9290-7563
ResearcherID: -GQB-3684-2022

**Tsybko Hanna**
Candidate of Pedagogic Sciences (Ph. D.), Associate Professor
Department of Computer Science and Computer Engineering
Taras Shevchenko National University “Chernihiv Colehium”
st. Hetman Polubotko, 53, Chernihiv, Ukraine, 14000
Email: a.tsb@ukr.net
ORCID: 0000-0002-1861-3003
ResearcherID: -AAC-6021-2021

**Shakotko Viktor**
Candidate of Pedagogic Sciences (Ph. D.), Associate Professor
Department of Technological and Professional Education
Oleksandr Dovzhenko Hlukhiv National Pedagogical University
Kyivska St., 24, Glukhiv, Sumy region, Ukraine, 41400
E-mail:vv0304@gmail.com
ORCID: 0000-0002-3004-5045
ResearcherID: -AAC-6021-2021