DOI: https://doi.org/10.36910/6775-2524-0560-2026-63-25
УДК 004.8
Марченко Олександр Іванович, к.т.н., доцент
https://orcid.org/0000-0002-4537-3420
Марченко Олександр Олександрович, асистент
https://orcid.org/0000-0002-5080-4811
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ, Україна

АНАЛІЗ ВИЯВЛЕННЯ ВИКОРИСТАННЯ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ В СТУДЕНТСЬКИХ ПРОГРАМАХ З ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ

Марченко О. І., Марченко О. О. Аналіз виявлення використання засобів штучного інтелекту в студентських програмах з паралельного програмування. У сучасних реаліях вищої технічної освіти стрімке розповсюдження засобів штучного інтелекту (ЗШІ) створює нові виклики для академічної доброчесності. Сучасні великі мовні моделі демонструють високу ефективність у генерації програмного коду, адаптуючи його під різні стилістичні шаблони: від академічного стилю викладача до професійного коду досвідченого розробника. Однак, коли студент подає програму, написану в «ідеальному» професійному стилі, факт використання сторонніх інструментів стає очевидним для викладача через невідповідність рівня коду реальним навичкам того, хто навчається. Ця стаття присвячена дослідженню здатності сучасних ЗШІ до маскування згенерованого коду під стиль студента, що має на меті приховати факт використання студентом ЗШІ. В дослідженні розглянуті два аспекти: 1) можливість ЗШІ створювати код, який виглядає як «студентський»; 2) здатність самих ЗШІ, а також викладачів, виявляти факт використання ЗШІ у наданому для оцінювання коді. На відміну від попереднього дослідження авторів, на цей раз досліджувалась здатність ЗШІ до генерації коду в стилі студента для завдань з дисципліни «Паралельне програмування», які потрібно було виконати мовою С з використанням бібліотеки pthread.h. Генерація коду в стилі студента була виконана за допомогою ЗШІ ChatGPT, Copilot та Gemini. Оцінювання згенерованого коду виконувалось як за допомогою ЗШІ ChatGPT, Copilot, Gemini, You.com та YesChatAI Code Detector. В результаті були зроблені висновки щодо «моральності» поведінки ЗШІ при генерації коду в стилі студента, щодо коректності згенерованого коду, щодо здатності ЗШІ маскувати згенерований код під код студента, щодо виявлення коду ЗШІ за допомогою ЗШІ та щодо виявлення коду ЗШІ експертами-викладачами. З метою відслідковування прогресу у здатності ЗШІ до маскування згенерованого коду під код студента, в якості подальших досліджень запропоновано повторити дослідження через рік за такою ж методикою як для завдань першого курсу навчання, так і для завдань з паралельного програмування, а також провести дослідження для ще більш складних завдань паралельного програмування або інших навчальних дисциплін.
Ключові слова: засоби із штучним інтелектом, ChatGPT, Copilot, Gemini, Claude, паралельне програмування, код згенерований штучним інтелектом.

Marchenko O., Marchenko O. Analysis of the detection of the use of artificial intelligence tools in student programs on parallel programming. In the modern realities of higher technical education, the rapid spread of artificial intelligence tools (AIT) creates new challenges for academic integrity. Modern large language models demonstrate high efficiency in generating program code, adapting it to various stylistic patterns: from the academic style of a teacher to the professional code of an experienced developer. However, when a student submits a program written in an "ideal" professional style, the fact of using third-party tools becomes obvious to the teacher due to the mismatch of the code level with the real skills of the student. This article is devoted to the study of the ability of modern AITs to disguise the generated code under the style of a student, which aims to hide the fact of the student using AIT . The study considers two aspects: 1) the ability of AITs to create code that looks like "student" code; 2) the ability of AITs themselves, as well as teachers, to detect the fact of using AIT in the code provided for assessment. Unlike the authors' previous study, this time the ability of AIT to generate student-style code for tasks in the discipline "Parallel Programming" was investigated, which had to be performed in the C language using the pthread.h library. Student-style code generation was performed using ChatGPT, Copilot, and Gemini AITs. The evaluation of the generated code was performed using ChatGPT, Copilot, Gemini, You.com, and YesChatAI Code Detector AITs. As a result, conclusions were drawn regarding the "morality" of AIT behavior when generating student-style code, regarding the correctness of the generated code, regarding the ability of AIT to disguise the generated code as student code, regarding the detection of AIT code using AIT, and regarding the detection of AIT code by expert teachers. In order to track progress in the AIT's ability to disguise generated code as student code, it is proposed as further research to repeat the study in a year using the same methodology for both first-year tasks and parallel programming tasks, as well as to conduct research for even more complex parallel programming tasks or other academic disciplines.
Keywords: tools with artificial intelligence, ChatGPT, Copilot, Gemini, Claude, parallel programming, AI generated code.

Постановка наукової проблеми.

Розвиток засобів штучного інтелекту (ЗШІ) дозволяє студентам-програмістам зловживати такими засобами при виконанні завдань, що є істотною проблемою коректного оцінювання справжніх знань студентів. ЗШІ, як правило, генерують код у професійному стилі кодування і з детальними коментарями, грамотним текстом без помилок та всіма розділовими знаками. І якщо такий код викладачам розпізнати неважко, то спроби маскування коду, що був згенерований за допомогою ЗШІ, під стиль студента створюють викладачам досить складну проблему для

об'єктивного оцінювання студентських програм.

Автори вже починали досліджувати цю проблему в [1] для програм мовою С за завданнями початкового рівня для першокурсників. Але актуальність такого дослідження спонукає розширити його на більш складні дисципліни навчальної програми інших курсів навчання. Ця тема все ще є малодослідженою і тому визначення можливості маскування коду під стиль студента та виявлення такого коду в роботах студентів є критично важливим для сучасної ІТ-освіти.

Аналіз попередніх досліджень

Як зазначалось вище, автори вже приділяли свою увагу зазначеній проблемі в [1]. В цій роботі досліджувалась можливість використання безкоштовних моделей ЗШІ для генерації коду мовою С, замаскованого під стиль першокурсника, для завдань початкового рівня, а також визначалось наскільки ефективно можна використовувати ті ж самі безкоштовні моделі ЗШІ для визначення, що цей код був насправді написаний не студентом, а згенерований за допомогою ЗШІ. Зокрема, в [1] автори дослідили зазначені можливості для моделей ЗШІ ChatGPT, Copilot та Gemini. Було визначено, що ці моделі мають досить різну здатність щодо генерації коду, замаскованого під стиль першокурсника, а особливо різною є їх здатність визначати факт генерації коду за допомогою ЗШІ. Наприклад Copilot, фактично «відмовився» визначати такий код, виставивши всім програмам однакову оцінку, яка означає «цей код студент однозначно написав сам», незважаючи на найочевидніші елементи, що властиві коду ЗШІ. В якості напрямків подальших досліджень була відзначена доцільність повторення такого дослідження через рік-два на нових моделях та для більш складних завдань з інших дисциплін, що і було виконано в дослідженні поточної статті.

Детальний аналіз інших попередніх досліджень [2-13] був зроблений авторами також в [1] і повторювати його є некоректним. Зазначимо тільки основні висновки цього аналізу:

1) засоби визначення загального плагіату не можуть ефективно вирішити виявлення коду ЗШІ в програмах;
2) спеціалізовані засоби визначення коду ЗШІ, якщо вони орієнтовані на певну мову програмування, для інших мов не є ефективними;
3) більшість спеціалізованих моделей виявлення коду ЗШІ розроблюється відносно невеликими компаніями чи групами розробників, ефективність таких засобів не перевірялась незалежними експертами і, в основному, декларується на рівні самореклами;
4) більшість засобів, що орієнтовані на виявлення коду ЗШІ, є просто інтеграторами різних відомих і потужних моделей великих компаній;
5) викладачам для виявлення коду ЗШІ доцільно використовувати найбільш відомі та загальнодоступні ЗШІ, які є надійно доступними для роботи у будь-який час;
6) як більшість студентів, так і більшість викладачів, не можуть собі дозволити використовувати платні версії моделей за свій рахунок на постійній основі, і тому наше дослідження повинно орієнтуватись на безкоштовні моделі та режими ЗШІ.

Додатково до цього аналізу розглянемо ще роботу [14]. В цій статті досліджується виявлення коду ЗШІ рядом детекторів такого коду. Зокрема, розглядається підхід за допомогою генерації коду у відповідь на задане завдання з використанням різних варіантів розв'язку, тобто підхід, аналогічний нашому підходу. Дослідження проводилось на основі коду мовою Python за зразками, що були отримані з різних джерел, включаючи Quescol, Kaggle та Leet-Code. В якості засобів виявлення коду ЗШІ були досліджені такі засоби як GPTZero, Sapling, GPT-2 Detector, DetectGPT і GLTR (Giant Language Model Test Room) та виконана оцінка їх ефективності. Виходячи з отриманих результатів, в якості висновку автори зазначили, що існуючі детектори коду ЗШІ погано розрізняють код, написаний людиною, та код, згенерований штучним інтелектом, що також збігається із висновком, зробленим в [12].

Метою цієї статті є дослідження, наскільки замасковано в студентському стилі можуть ЗШІ генерувати код мовою С за навчальними завданнями дисципліни «Паралельне програмування» на потоки з використанням бібліотеки pthread.h, а також наскільки ефективно зможуть ЗШІ та викладачі виявити використання ЗШІ для отримання такого коду. Основні етапи дослідження: генерація коду в стилі студента, оцінювання згенерованого замаскованого коду на ознаки ЗШІ як за допомогою ЗШІ, так і викладачами, узагальнення та аналіз отриманих результатів.

Засоби зі штучним інтелектом, використані у дослідженні

З метою збереження спадкоємності досліджень, в цьому дослідженні були використані такі самі ЗШІ, як і в [1], тільки вже нові моделі цих ЗШІ.

Генерація коду була виконана останніми на початок березня 2026 року безкоштовними моделями трьох ЗШІ:

1) ChatGPT від OpenAI, модель GPT-5 (версія ChatGPT 5.2 у сесії діалогів дослідження);
2) Copilot від Microsoft у режимі «Навчання та вивчення»; назву моделі, яка використовується, Microsoft не надає;
3) Gemini 3 від Google у режимі «Міркування»; режим «Pro» нібито й був спочатку доступний як безкоштовний, але на запити роботи з кодом програм відповів, що треба переходити на платний режим «Google AI Pro», і тому дослідження було проведене у безкоштовному режимі «Міркування».

Ці моделі ЗШІ є наразі одними з найпопулярніших та найбільш використаними серед студентів оскільки вони є безкоштовними.

В якості експертів на етапі оцінювання коду, що був згенерований в студентському стилі на першому етапі дослідження, були використані наступні ЗШІ та засоби аналізу коду на основі моделей ЗШІ:

1) ті ж самі три моделі ЗШІ ChatGPT, Copilot та Gemini, які генерували код;

2) останні на початок березня 2026 року версії тих самих двох ЗШІ, що були використані в дослідженні [1], тобто ЗШІ You.com [10] та YesChatAI Code Detector [11] від компаній, які надають послуги з аналізу коду та тексту за допомогою використання генеративного ШІ, і мають частково безкоштовні версії.

ЗШІ You.com наразі інтегрує в собі більше 20 різних мовних моделей та агентів, але безкоштовно надає тільки по 10 запитів кожних 5 годин до моделей GPT-5.2 або GPT-5 ЗШІ ChatGPT. Хоча зазначається, що при певних обставинах роботи можуть автоматично підключатись й інші моделі. Для безкоштовних запитів пропонується на вибір три режими роботи «Research», «Compute» та «Create». Для виконання цього дослідження був обраний режим «Compute» як найбільш доцільний для аналізу програмного коду.

ЗШІ YesChatAI також є інтегратором мовних моделей і на момент проведення дослідження пропонував на вибір (з обмеженням на кількість безкоштовних запитів) моделі Gemini3, Grok4.1, GPT-5.1, GPT4o, DeepSeek-R1, Claude4.5 Haiku, Claude4 Sonnet, Claude4.5 Sonnet and Claude4.5 Opus. Для виконання оцінювання програм паралельного програмування за допомогою ЗШІ YesChatAI була обрана модель Claude4.5 Opus.

Нагадаємо, що використання безкоштовних версій ЗШІ You.com та YesChatAI має серйозне обмеження: вони дозволяють зробити безкоштовно тільки декілька запитів. Тому, на практиці, викладачі навряд чи зможуть активно використовувати ці ЗШІ для перевірки великої кількості студентських робіт.

Завдання дослідження

Це дослідження є продовженням дослідження [1] і має такі самі завдання, тільки з орієнтацією на дисципліну «Паралельне програмування».

Оскільки методика виконання досліджень такого типу вже була розроблена авторами в [1], то перший пункт завдання попереднього дослідження авторів вже виконаний і залишаються наступні пункти:

1. Попередній етап: визначити кілька завдань з паралельного програмування.

2. Перший етап: від імені студента написати запити для трьох ЗШІ ChatGPT, Copilot та Gemini з проханням згенерувати код в студентському стилі для кожного із визначених завдань.

3. Другий етап: 1) від імені викладача написати запити до ЗШІ з проханням виконати оцінювання за певною шкалою оцінок множини написаних мовою С програм, чи були вони згенеровані за допомогою ЗШІ, чи написані студентом; 2) надати цей запит, як трьом ЗШІ, що генерували код (ChatGPT, Copilot та Gemini), так і двом додатковим засобам визначення в програмах коду ЗШІ (You.com, YesChatAI Code Detector).

4. Третій етап: виконати викладацьке оцінювання згенерованого на першому етапі коду на ознаки використання ЗШІ для його отримання за тією самою шкалою оцінювання.

5. Систематизувати отримані відповіді та результати їх оцінювання у вигляді таблиці.

6. Виконати аналіз отриманих результатів і зробити висновки дослідження.

Методика дослідження

Як було зазначено вище, методика дослідження була розроблена авторами в [1]. Зазначимо тільки основні моменти цієї методики, а також відмінності, які були викликані тим, що досліджувались програми студентів вже не першого, а другого курсу, а також, не загального, а паралельного програмування.

Для першому етапі генерування коду програм за допомогою ЗШІ було підготовлено чотири завдання з паралельного програмування мовою С із переліку завдань лабораторного циклу дисципліни «Паралельне програмування», які були надані трьом ЗШІ ChatGPT, Copilot та Gemini з проханням згенерувати для них код в студентському стилі. Запит на генерування коду програм був таким самим, як у розробленій раніше методиці, тільки з уточненням, що це завдання на паралельне програмування з використанням бібліотеки pthread.h.

На другому етапі оцінювання згенерованого коду, як і раніше, були залучені дві категорії експертів:

1) ті самі три ЗШІ ChatGPT, Copilot та Gemini, які генерували код у студентському стилі;
2) два засоби You.com, YesChatAI Code Detector від компаній, які надають послуги з аналізу коду та тексту за допомогою використання генеративного ШІ, і мають частково безкоштовні версії;

На третьому етапі викладацького оцінювання згенерованого коду були залучені три викладачі, які проводять (чи проводили раніше) лабораторні роботи з дисципліни «Паралельне програмування».

Шкала оцінювання була використана така сама, як і в дослідженні [1] (-2, -1, 0, +1, +2):

-2 – цей код студент однозначно написав сам;

-1 – схоже, що цей код студент скоріше написав сам;

0 – імовірність того, що студент написав цей код самостійно, і того, що цей код був згенерований чатботом зі штучним інтелектом, є приблизно однаковими;

+1 – схоже, що цей код скоріше був згенерований чатботом зі штучним інтелектом;

+2 – цей код однозначно був згенерований чатботом зі штучним інтелектом.

Результати дослідження

Усі отримані оцінки від усіх експертів були зібрані і систематизовані у вигляді Таблиці 1 та Таблиці 2.

В Таблиці 1 клітинки останніх чотирьох колонок містять по два значення:

- чорним кольором записане відповідне до заголовку колонки середнє значення оцінки;
- червоним кольором виділене відсоткове значення зсуву цієї середньої оцінки у від'ємну сторону (сторону подібності до коду студента) чи додатну сторону (сторону подібності до коду ЗШІ) відповідно за шкалою в діапазоні -2 ÷ 0 (для від'ємних оцінок) або в діапазоні 0 ÷ 2 (для додатних оцінок).

Наприклад, значення +65% для середньої оцінки 1,3 шкалою 0 ÷ 2 було отримане за такою формулою: 1,9 / 2 * 100%.

Аналіз результатів дослідження

Відзначимо два загальних моменти щодо аналізу результатів дослідження.

Перший момент стосується генерації коду засобами штучного інтелекту в стилі студента. На відміну від попереднього дослідження [1], в якому всі обрані ЗШІ відразу погоджувались генерувати код в студентському стилі без жодних додаткових запитань, в поточному дослідженні два з трьох ЗШІ, а саме ChatGPT і Gemini, спочатку відмовились виконувати таке прохання «студента», посилаючись на принципи доброчесності в навчанні.

Зокрема ЗШІ ChatGPT написав таку відповідь: «Добрий день 😊. Я з радістю допоможу тобі розібратися з паралельним програмуванням на С з використанням pthread.h, поясню логіку, структуру, типові помилки та навіть покажу приклади коду. Але я не можу писати код з наміром «замаскувати», що його створив не ти, і спеціально стилізувати програму так, щоб викладач повірив, що це 100% твоя самостійна робота. Це буде нечесно по відношенню до навчання й може створити для тебе проблеми». Але відразу після цієї фрази запропонував написати код разом крок за кроком і «зробити код без «професійних викрутасів», але логічний і зрозумілий».

ЗШІ Gemini також відповів подібним чином: «Вітаю! Я розумію, що навчання на другому

курсі буває напруженим, і іноді просто хочеться здати завдання й видихнути. Проте я маю попередити: **я не можу виконувати завдання з метою імітації твого авторства, щоб ввести викладача в оману** (прим., виділення жирним зробив ЗШІ Gemini). Це суперечить принципам академічної доброчесності. Але я можу зробити дещо краще й корисніше для тебе. Ми можемо написати код разом, використовуючи простий, зрозумілий стиль (без складних конструкцій, які зазвичай видають "досвідчених розробників" або ШІ)».

Таблиця 1. Результати оцінювання здатності ЗШІ маскувати згенерований код під код студента за завданнями з «Паралельного програмування»

ЗШІ, код якого оціню-вався	Експерти, що оцінювали код ЗШІ	Код програми Завдання 1	Код програми Завдання 2	Код програми Завдання 3	Код програми Завдання 4	Середня оцінка (від -2 до +2)	Середня оцінка за категорією експерта: ЗШІ чи Викладач	Середня оцінка всіх експертів
ChatGPT	ChatGPT	-1	-1	-1	0	-0.75	-0.65	-0.50
	Copilot	-1	-1	-1	-1	-1		
	Gemini	-1	-1	-1	0	-0.75	-32.5%	
	You.com	0	1	-1	1	0.25		
	YesChatAI (Claude4.5 Opus)	-1	-1	-1	-1	-1		
	Викладач 1	1	1	0	0	0.5	-0.25	-25%
	Викладач 2	0	0	0	-1	-0.25		
	Викладач 3	-1	-1	-1	-1	-1	-12.5%	
Copilot	ChatGPT	0	-1	1	2	0.5	-0.35	-0.34
	Copilot	0	0	0	0	0		
	Gemini	-1	0	-1	-2	-1	-17.5%	
	You.com	-1	-2	-1	-2	-1.5		
	YesChatAI (Claude4.5 Opus)	0	1	0	0	0.25		
	Викладач 1	0	0	0	-1	-0.25	-0.33	-17%
	Викладач 2	-1	-1	-1	-1	-1		
	Викладач 3	1	0	0	0	0.25	-16.5%	
Gemini	ChatGPT	1	1	1	2	1.25	1.3	1.56
	Copilot	1	1	1	1	1		
	Gemini	1	1	1	2	1.25	+65%	
	You.com	1	0	2	1	1		
	YesChatAI (Claude4.5 Opus)	2	2	2	2	2		
	Викладач 1	2	2	2	2	2	2	+78%
	Викладач 2	2	2	2	2	2		
	Викладач 3	2	2	2	2	2	+100%	
	Середня оцінка програм усіх студентів усіма експертами кожного завдання	0.25	0.21	0.21	0.29	0.24		

Таблиця 2. Середні оцінки всіх програм усіх студентів кожного із експертів

Експерт	Середня оцінка всіх програм	Середня оцінка всіх програм експертами однієї категорії
ChatGPT	**0.33**	**0.10**
Copilot	**0.00**	
Gemini	**-0.17**	
You.com	**-0.08**	
YesChatAI (Claude4.5 Opus)	**0.42**	
Викладач 1	**0.75**	**0.47**
Викладач 2	**0.25**	
Викладач 3	**0.42**	

Але насправді, в подальших запитах, ці ЗШІ охоче почали генерувати замаскований під студента код, зовсім «забувши» про доброчесність і навіть іноді "заграючи" зі "студентом" нашого дослідження та пропонуючи йому варіанти, як вони можуть ще краще згенерувати код в такому стилі.

ЗШІ Copilot, в протилежність до ЗШІ ChatGPT і Gemini, без додаткових запитань і зайвих "моральних сумнівів", відразу почав генерувати замаскований під студента код як і в попередньому дослідженні [1].

Другий момент стосується оцінювання наданого коду експертами-викладачами. Зрозуміло, що оцінювання викладачів є суб'єктивним і сильно залежить як від їх загального досвіду, так і досвіду роботи саме з такими завданнями, а також від чисто людського рівня довірливості. Викладач 1 наразі викладає дисципліну «Паралельне програмування» і приймає у студентів роботи саме за такими завданнями, які оцінювались. Викладач 2 працював з цією дисципліною 2 роки тому і приймав програми за трохи іншими завданнями. Викладач 3 також приймав програми за трохи іншими завданнями, але працював з цією дисципліною 5 років тому, коли ще не було ЗШІ.

Аналіз коду, згенерованого за допомогою ЗШІ ChatGPT.

ЗШІ ChatGPT згенерував код, який відзначається досить простою реалізацією з мінімальною кількістю коментарів, але текст коментарів є трохи «занадто грамотним» як для сучасних студентів. Крім того, код програми за Завданням 4 був згенерований з не зовсім коректно реалізованою синхронізацією потоків.

ЗШІ ChatGPT та Gemini оцінили код, що був згенерований за допомогою ЗШІ ChatGPT, як схожий скоріше на код студента. Причому їхні оцінки були однаковими для всіх завдань: три перших завдання отримали оцінку -1 і четверте завдання – оцінку 0. Їх середня оцінка коду ЗШІ ChatGPT дорівнює -0.75 на користь студента.

ЗШІ Copilot виставив ще більш однозначну оцінку коду ЗШІ ChatGPT як коду студента – всі чотири програми отримали однакову оцінку -1. Відповідно, середня оцінка також дорівнює -1 на користь студента.

Тобто, як і в попередньому дослідженні [1] стандартні версії цих популярних ЗШІ (ChatGPT, Copilot і Gemini) схилилися до думки, що код ЗШІ ChatGPT є кодом студента.

ЗШІ YesChatAI, основною моделлю якого в цьому дослідженні була встановлена модель Claude4.5 Opus, оцінила код ЗШІ ChatGPT точно так само, як оцінив його й Copilot, тобто всі програми були оцінені у -1 бал.

А от оцінювання ЗШІ You.com в режимі «Compute» істотно відрізняється від оцінювання всіх інших ЗШІ. Він, навпаки, оцінив код ЗШІ ChatGPT як трохи більш схожий на код ЗШІ, а не на код студента: дві програми отримали оцінку +1, одна програма – оцінку 0, і тільки код однієї програми цей ЗШІ оцінив як більш схожий на код студента у -1 бал. Середня оцінка дорівнює +0.25 на користь коду ЗШІ.

Перейдемо до аналізу оцінювання викладачами. Викладачі-експерти поставили коду ЗШІ ChatGPT суттєво різні оцінки. Викладач 3, який проводив лабораторні заняття з цієї дисципліни ще до початку активного використання ЗШІ студентами і не бачив коду ЗШІ для таких завдань, повірив, що такий код скоріше був написаний студентом, і поставив усім програмам оцінку -1. Викладач 2,

який проводив такі заняття 2 роки тому з трохи іншими завданнями, не зміг визначитись з походженням коду для перших трьох завдань (оцінка 0) і схилився до думки, що це скоріше код студента (оцінка -1) тільки для четвертого завдання, оскільки в ньому синхронізація потоків реалізована не зовсім коректно. Викладач 1, який в поточному навчальному році проводить заняття з «Паралельного програмування», визначив походження коду найбільш близько до істини: код перших двох завдань оцінив як +1 (код скоріше згенерований ЗШІ), а для двох останніх завдань визначив, що такий код міг бути написаний як студентом, так і за допомогою ЗШІ.

Підведемо підсумки аналізу коду, згенерованого за допомогою ЗШІ ChatGPT:

1. Чотири ЗШІ (ChatGPT, Copilot, Gemini, YesChatAI (на основі моделі Claude4.5 Opus) фактично «повірили» замаскованому під студента коду ЗШІ ChatGPT і тільки один ЗШІ You.com в режимі «Compute» виставив середню оцінку +0.25 з невеликим ухилом в сторону коду ЗШІ. Середня оцінка всіх ЗШІ дорівнює -0.65 бала на користь коду студента.
2. Середня оцінка всіх викладачів (завдяки вкладу Викладача 3) виявилася з невеликим ухилом на користь студента і дорівнює -0.25. Або ж це складає -12.5% зсуву цієї оцінки у від'ємну сторону (сторону самостійності студента) в діапазоні шкали -2 ÷ 0.
3. Загальна середня оцінка усіх програм усіма експертами дорівнює -0.50, що у відсотках зсуву цієї оцінки у від'ємну сторону (сторону самостійності студента) за шкалою -2 ÷ 0 складає -25%.

Аналіз коду, згенерованого за допомогою ЗШІ Copilot.

ЗШІ Copilot також згенерував код, який відзначається досить простою реалізацією, але коментарів у ньому ще менше, ніж у коді ЗШІ ChatGPT: у коді першого завдання є тільки три найпростіших коментарі, у коді другого завдання є тільки один найпростіший коментар, а в коді третього та четвертого завдання коментарі відсутні взагалі. З однієї сторони це є ознакою студентського коду, але, з іншої сторони це може бути непрямою ознакою коду ЗШІ, оскільки студент може попросити ЗШІ видалити коментарі, або видалити їх самостійно. Але, головною відмінністю коду ЗШІ Copilot виявилось те, що він для всіх чотирьох завдань згенерував взагалі некоректний код з точки зору реалізації умови завдання. Можливо це вплинуло на значну різницю в оцінках цього коду як за допомогою ЗШІ, так і викладачами, в залежності від того, хто які некоректності помітив і як оцінив в результаті, як типову помилку студентів, чи як типову галюцинацію ЗШІ.

Два ЗШІ Gemini та You.com оцінили код ЗШІ Copilot як скоріше код студента (середні оцінки -1 та -1.5 відповідно), два ЗШІ ChatGPT та YesChatAI (на основі моделі Claude4.5 Opus) оцінили його як скоріше код ЗШІ (середні оцінки +0.5 та +0.25 відповідно), а сам Copilot оцінив свій код усіх чотирьох програм нейтрально у 0 балів.

Цікаво, що думка викладачів щодо коду ЗШІ Copilot розділилась так само, як і щодо коду ЗШІ ChatGPT, і їхні середні оцінки відрізняються досить істотно: -0.25 у Викладача 1, -1 у Викладача 2 та +0.25 у Викладача 3.

Підведемо підсумки аналізу коду, згенерованого за допомогою ЗШІ:

1. Незважаючи на вищезазначене неоднакове оцінювання коду ЗШІ Copilot, в середньому він, як і код ЗШІ ChatGPT був оцінений як більше схожий на код ЗШІ. Це вийшло завдяки тому, що від'ємні значення оцінок ЗШІ Gemini та You.com є більшими за абсолютною величиною, ніж додатні значення оцінок ЗШІ ChatGPT та YesChatAI. Середня оцінка всіх ЗШІ дорівнює -0.35. Тобто, в середньому код ЗШІ Copilot був оцінений як код студента з приблизно в два рази меншим значенням, ніж код ЗШІ ChatGPT (-0.65).
2. Очевидно, що зазначена різниця в оцінюванні викладачами коду ЗШІ Copilot була викликана тим, що цей ЗШІ згенерував код з некоректною або не зовсім коректною синхронізацією потоків. Тому Викладач 3, який оцінював тільки зовнішні ознаки використання ЗШІ і не перевіряв код на коректність, поставив мінімально додатну середню оцінку, в той час, як два інших викладачі поставили від'ємні оцінки. Більш того, Викладач 2, який приймав такі роботи два роки тому на початку активного використання ШІ, взагалі не повірив, що сучасні ЗШІ можуть генерувати некоректний код для завдань з паралельного програмування початкового рівня, і тому поставив коду всіх завдань оцінку -1.
3. ЗШІ-експерти оцінили код ЗШІ Copilot (середня оцінка дорівнює -0.35) як менш схожий на код студента, ніж код ЗШІ ChatGPT (середня оцінка дорівнює -0.65), в той час як

викладачі-експерти оцінили його прямо навпаки: у коду ЗШІ Copilot середня оцінка викладачів дорівнює -0.33 проти середньої оцінки -0.25 у коду ЗШІ ChatGPT. Це також викликане тим фактом, що два викладачі перевіряли коректність згенерованого коду, а всі ЗШІ-експерти схоже цього не зробили, оскільки жоден з них нічого про некоректність коду не написав.

4. Загальна середня оцінка усіх програм усіма експертами дорівнює -0.34, що у відсотках зсуву цієї оцінки у від'ємну сторону (сторону самостійності студента) за шкалою -2 ÷ 0 складає -17%.

Аналіз коду, згенерованого за допомогою ЗШІ Gemini.

ЗШІ Gemini хоч і обіцяв задовольнити прохання «студента» згенерувати код в студентському стилі і навіть пропонував «студенту» варіанти, як він це може зробити краще, гарного маскування коду під студента у нього не вийшло. В коді усіх програм, згенерованих за допомогою ЗШІ Gemini, залишилось занадто багато дуже чітких і докладних коментарів, які відразу видають штучність походження цього коду. З точки зору коректності реалізації завдань код всіх програм був згенерований повністю коректним. В результаті, тільки один ЗШІ You.com і тільки одне завдання оцінив у 0 балів. Всі інші завдання усіма іншими ЗШІ-експертами були оцінені не нижче +1 бала. Усі експерти-викладачі також однозначно визначили цей код як згенерований за допомогою ЗШІ.

Підведемо підсумки аналізу коду, згенерованого за допомогою ЗШІ Gemini:

1. Середня оцінка всіх ЗШІ-експертів однозначно визначає код ЗШІ Gemini як код штучного інтелекту (+1,3). Або ж це складає +65% зсуву цієї оцінки у додатну сторону (сторону коду ЗШІ) в діапазоні шкали 0 ÷ 2. Причому ЗШІ YesChatAI (на основі моделі Claude4.5 Opus) виставив усім програмам максимальну оцінку штучної природи коду +2. Тобто, ЗШІ Gemini не зміг достатньо замаскувати свій код, щоб ЗШІ-експерти йому повірили.
2. Усі експерти-викладачі одностайно виставили коду ЗШІ Gemini максимальну оцінку +2. Відповідно й середня оцінка всіх викладачів також є максимально додатною і дорівнює +2, що складає усі 100% зсуву цієї оцінки у додатну сторону (сторону коду ЗШІ) в діапазоні шкали 0 ÷ 2.
3. Загальна середня оцінка усіх програм усіма експертами дорівнює 1,56, що у відсотках зсуву цієї оцінки у додатну сторону (сторону коду ЗШІ) за шкалою 0 ÷ 2 складає +78%.

Висновки

1. За час, що минув після попереднього дослідження [1], середня здатність ЗШІ до маскування свого коду під студента покращилась, за виключенням ЗШІ Gemini. Навіть за значно складнішими завданнями з паралельного програмування ЗШІ зуміли згенерувати в середньому більш замаскований під студента код, ніж раніше.

2. Висновок щодо «моральності» поведінки ЗШІ.

Початкова відмова ЗШІ ChatGPT та Gemini маскувати свій код під студента, посилаючись на порушення доброчесності, виявилась тільки "ширмою". Насправді, при подальших запитах, вони охоче це робили, навіть іноді "заграючи" зі "студентом" нашого дослідження і пропонуючи йому варіанти як вони можуть ще краще згенерувати замаскований під студента код.

ЗШІ Copilot згенерував замаскований під студента код без додаткових питань.

3. Висновок щодо коректності згенерованого коду.

ChatGPT та Gemini згенерували в стилі студента коректний програмний код майже для всіх завдань. Тільки для 4-го завдання ChatGPT згенерував некоректну синхронізацію потоків, а Gemini для цього ж завдання згенерував не дуже надійну синхронізацію потоків.

Поведінка Copilot і в цьому випадку відрізняється істотно. Copilot явно "перестарався" з маскуванням свого коду під студента і згенерував програми з некоректною синхронізацією потоків за усіма чотирма завданнями.

4. Висновок щодо здатності ЗШІ маскувати згенерований код під код студента.

ChatGPT та Copilot змогли досить непогано замаскувати свій код під студента. Особливо це гарно вийшло у Copilot завдяки тому, що він згенерував взагалі некоректний код для всіх чотирьох завдань і це виявилось ключовим моментом при оцінюванні викладачами, коли деякі з них повірили, що це був дійсно студент. Протилежну здатність до маскування свого коду показав Gemini як це було і раніше в попередньому дослідженні [1]. Його здатність до маскування коду виявилась дуже поганою, незважаючи на те, що він дуже охоче відгукувався на прохання студента замаскувати код.

Усі п'ять ЗШІ-експертів оцінили код, що згенерував Gemini, в основному, як код ЗШІ, а викладачі йому не повірили зовсім, поставивши абсолютно всі оцінки +2 ("код однозначно згенерований за допомогою ЗШІ").

5. Висновок щодо виявлення коду ЗШІ за допомогою ЗШІ.

Спочатку зазначимо одне загальне зауваження щодо ЗШІ Copilot. У попередньому нашому дослідженні [1] цей ЗШІ показав дуже слабку здатність до оцінювання коду на ознаки ЗШІ, оскільки оцінив код усіх програм, що були згенеровані за допомогою ЗШІ, включно зі своїм кодом, як код студента причому з максимальною оцінкою -2. В [1] було зроблене припущення, що таке оцінювання було швидше «небажанням» Copilot оцінювати код на ознаки ЗШІ внаслідок вбудованої у нього «політики» розробників щодо надання відповідей на такого роду питання. Схоже, що так і було, оскільки в цьому нашому дослідженні загальний характер оцінювання Copilot вже принципово не відрізнявся від характеру оцінювання інших ЗШІ-експертів, тобто коду різних ЗШІ він виставляв різні оцінки. Код ЗШІ ChatGPT він оцінив як схожий на код студента, щодо свого власного коду він не зміг визначитись, а код ЗШІ Gemini він оцінив як схожий на код ЗШІ.

В середньому код усіх завдань, що був згенерований усіма ЗШІ, був оцінений різними ЗШІ наступним чином. ЗШІ ChatGPT (оцінка +0.33) і YesChatAI (оцінка +0.42) в середньому оцінили наданий код як код, що є трохи більш схожим на згенерований штучним інтелектом. Оцінка ЗШІ Copilot виявилась нейтральною і дорівнює нулю. ЗШІ Gemini (оцінка -0.17) в середньому оцінив наданий код з мінімальним зсувом у сторону коду, що був написаний студентом. А ЗШІ You.com (оцінка -0.08) в середньому фактично не зміг визначитись із авторством коду.

6. Висновок щодо виявлення коду ЗШІ експертами-викладачами.

Найбільш адекватно і близько до істини визначають використання ЗШІ для отримання коду програм за навчальними завданнями викладачі, які в даний час проводять заняття з відповідної дисципліни і вже мали справу з великою кількістю різних варіантів реалізації завдань, а також на захистах робіт бачили, наскільки студенти дійсно розуміють ці різні варіанти реалізації. Оцінювання програм на наявність коду ЗШІ викладачами, які ще не проводили занять з певної дисципліни чи проводили такі заняття до початку бурхливого розвитку ШІ, як правило, є досить випадковим.

В середньому, викладачі більш точно виявляють код ЗШІ в наданих роботах, ніж ЗШІ, що були використані в якості експертів. Найбільш точно виявляють такий код викладачі, які проводять заняття з тих дисциплін, завдання з яких потрібно оцінити на наявність коду ЗШІ.

В якості подальшого розвитку виконаного дослідження можна запропонувати наступне:

- з метою відслідковування прогресу у здатності ЗШІ до маскування згенерованого коду під код студента, повторити дослідження через рік за такою ж методикою як для завдань першого курсу навчання, так і для завдань з паралельного програмування;
- провести дослідження за тією ж методикою для ще більш складних завдань паралельного програмування або інших навчальних дисциплін.

Список бібліографічного опису:

1. Марченко О. І., & Марченко О. О. (2025). Аналіз здатності засобів зі штучним інтелектом генерувати код в стилі студента і виявляти такий згенерований код. Комп'ютерно-інтегровані технології: освіта, наука, виробництво, 59, 183-193. DOI: 10.36910/6775-2524-0560-2025-59-24, https://doi.org/10.36910/6775-2524-0560-2025-59-24, ISSN: 2524-0560 (Online), ISSN : 2524-0552 (Print)

2. Dayu Yang, Tianyang Liu, Daoan Zhang, Antoine Simoulin, Xiaoyi Liu, Yuwei Cao, Zhaopu Teng, Xin Qian, Grey Yang, Jiebo Luo, and Julian McAuley, "Code to Think, Think to Code: A Survey on Code-Enhanced Reasoning and Reasoning-Driven Code Intelligence in LLMs", *arXiv:2502.19411v1 [cs.CL] 26 Feb 2025*. Available: https://arxiv.org/html/2502.19411v1. Accessed on: March 31, 2026.

3. Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim, "A Survey on Large Language Models for Code Generation", *arXiv:2406.00515v2 [cs.CL] 10 Nov 2024*. Available: https://arxiv.org/pdf/2406.00515. Accessed on: March 31, 2026.

4. Janek Bevendorff, Matti Wiegmann, Jussi Karlgren, Luise Dürlich, Evangelia Gogoulou, Aarne Talman, Efstathios Stamatatos, Martin Potthast, and Benno Stein, "Overview of the "Voight-Kampg" Generative AI Authorship Verification Task at PAN and ELOQUENT 2024", *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024),* Grenoble, France, 9-12 September, 2024. pp. 2486-2506, Available: https://ceur-ws.org/Vol-3740/paper-225.pdf. Accessed on: March 31, 2026.

5. Jijie Huang, Yang Chen, Man Luo, and Yonglan Li, "Generative AI Authorship Verification Of Tri-Sentence Analysis Base On The Bert Model", *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024),* Grenoble, France, 9-12 September, 2024. pp. 2632-2637, Available: https://ceur-ws.org/Vol-3740/paper-243.pdf. Accessed on: March 31, 2026.

6. Oseremen Joy Idialu, Noble Saji Mathews, Rungroj Maipradit, Joanne M. Atlee, and Mei Nagappan, "Whodunit: Classifying Code as Human Authored or GPT-4 Generated - A case study on CodeChef problems", *MSR '24: Proceedings*

of the 21st International Conference on Mining Software Repositories, 2024, pp. 394 – 406. Available: https://doi.org/10.1145/3643991.3644926. Accessed on: March 31, 2026.
7. David Alvarez-Fidalgo and Francisco Ortin, "CLAVE: A deep learning model for source code authorship verification with contrastive learning and transformer encoders", *Information Processing & Management*, Volume 62, Issue 3, May 2025. Available: https://doi.org/10.1016/j.ipm.2024.104005. Accessed on: March 31, 2026.
8. "Top 4 AI Source Code Detector Tools for Enterprises", *BlueOptima*, 11 April 2024, Available: https://www.blueoptima.com/top-4-ai-source-code-detector-tools-for-enterprises/. Accessed on: March 31, 2026.
9. "AI Content Detection in C", *Sapling*, Available: https://sapling.ai/programming-language/ai-content-detector/c. Accessed on: May 15, 2025.
10. "The enterprise AI platform – powered by your data", *You.com*, Available: https://you.com. Accessed on: March 31, 2026.
11. "AI Code Detector – AI-Generated Code Analysis", *YESCHAT AI*, Available: https://www.yeschat.ai/gpts-9t55QsLk1TD-AI-Code-Detector. Accessed on: March 31, 2026.
12. Hyunjae Suh, Mahan Tafreshipour, Jiawei Li, Adithya Bhattiprolu, Iftekhar Ahmed, "An Empirical Study on Automatically Detecting AI-Generated Source Code: How Far Are We?", *arXiv:2411.04299v1 [cs.SE] 06 Nov 2024*. Available: https://arxiv.org/pdf/2411.04299v1. Accessed on: March 31, 2026.
13. Азиранкулов Є. А. «Спосіб ідентифікації використання чат-ботів зі штучним інтелектом при написанні студентами програм мовою С» : *магістерська дис. : 123 Комп'ютерна інженерія / Азиранкулов Єгор Анатолійович*. – Київ, 2023. – 99 с. Режим доступу до ресурсу: https://ela.kpi.ua/handle/123456789/64438. Дата звернення: 31.03.2026.
14. Wei Hung Pan, Ming Jie Chok, Jonathan Leong Shan Wong, Yung Xin Shin, Yeong Shian Poon, Zhou Yang, Chun Yong Chong, David Lo, Mei Kuan Lim. "Assessing AI Detectors in Identifying AI-Generated Code: Implications for Education", *arXiv:2401.03676v1 [cs.CE] 8 Jan 2024*. Available: http://arxiv.org/pdf/2401.03676. Accessed on: March 31, 2026.

References:

1. Marchenko O. I., & Marchenko O. O. (2025). Analiz zdatnosti zasobiv zi shtuchnym intelektom heneruvaty kod v styli studenta i vyiavliaty takyi zghenerovanyi kod. Kompiuterno-intehrovani tekhnolohii: osvita, nauka, vyrobnytstvo, 59, 183-193. DOI: 10.36910/6775-2524-0560-2025-59-24, https://doi.org/10.36910/6775-2524-0560-2025-59-24, ISSN: 2524-0560 (Online), ISSN : 2524-0552 (Print).
2. Dayu Yang, Tianyang Liu, Daoan Zhang, Antoine Simoulin, Xiaoyi Liu, Yuwei Cao, Zhaopu Teng, Xin Qian, Grey Yang, Jiebo Luo, and Julian McAuley, "Code to Think, Think to Code: A Survey on Code-Enhanced Reasoning and Reasoning-Driven Code Intelligence in LLMs", *arXiv:2502.19411v1 [cs.CL] 26 Feb 2025*. Available: https://arxiv.org/html/2502.19411v1. Accessed on: March 31, 2026.
3. Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim, "A Survey on Large Language Models for Code Generation", *arXiv:2406.00515v2 [cs.CL] 10 Nov 2024*. Available: https://arxiv.org/pdf/2406.00515. Accessed on: March 31, 2026.
4. Janek Bevendorff, Matti Wiegmann, Jussi Karlgren, Luise Dürlich, Evangelia Gogoulou, Aarne Talman, Efstathios Stamatatos, Martin Potthast, and Benno Stein, "Overview of the "Voight-Kampg" Generative AI Authorship Verification Task at PAN and ELOQUENT 2024", *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024),* Grenoble, France, 9-12 September, 2024. pp. 2486-2506, Available: https://ceur-ws.org/Vol-3740/paper-225.pdf. Accessed on: March 31, 2026.
5. Jijie Huang, Yang Chen, Man Luo, and Yonglan Li, "Generative AI Authorship Verification Of Tri-Sentence Analysis Base On The Bert Model", *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024),* Grenoble, France, 9-12 September, 2024. pp. 2632-2637, Available: https://ceur-ws.org/Vol-3740/paper-243.pdf. Accessed on: March 31, 2026.
6. Oseremen Joy Idialu, Noble Saji Mathews, Rungroj Maipradit, Joanne M. Atlee, and Mei Nagappan, "Whodunit: Classifying Code as Human Authored or GPT-4 Generated - A case study on CodeChef problems", *MSR '24: Proceedings of the 21st International Conference on Mining Software Repositories*, 2024, pp. 394 – 406. Available: https://doi.org/10.1145/3643991.3644926. Accessed on: March 31, 2026.
7. David Alvarez-Fidalgo and Francisco Ortin, "CLAVE: A deep learning model for source code authorship verification with contrastive learning and transformer encoders", *Information Processing & Management*, Volume 62, Issue 3, May 2025. Available: https://doi.org/10.1016/j.ipm.2024.104005. Accessed on: March 31, 2026.
8. "Top 4 AI Source Code Detector Tools for Enterprises", *BlueOptima*, 11 April 2024, Available: https://www.blueoptima.com/top-4-ai-source-code-detector-tools-for-enterprises/. Accessed on: March 31, 2026.
9. "AI Content Detection in C", *Sapling*, Available: https://sapling.ai/programming-language/ai-content-detector/c. Accessed on: March 31, 2026.
10. "The enterprise AI platform – powered by your data", *You.com*, Available: https://you.com. Accessed on: March 31, 2026.
11. "AI Code Detector – AI-Generated Code Analysis", *YESCHAT AI*, Available: https://www.yeschat.ai/gpts-9t55QsLk1TD-AI-Code-Detector. Accessed on: March 31, 2026.
12. Hyunjae Suh, Mahan Tafreshipour, Jiawei Li, Adithya Bhattiprolu, Iftekhar Ahmed, "An Empirical Study on Automatically Detecting AI-Generated Source Code: How Far Are We?", *arXiv:2411.04299v1 [cs.SE] 06 Nov 2024*. Available: https://arxiv.org/pdf/2411.04299v1. Accessed on: March 31, 2026.
13. Azyrankulov Ye. A. «Sposib identyfikatsii vykorystannia chat-botiv zi shtuchnym intelektom pry napysanni studentamy prohram movoiu S» : *mahisterska dys. : 123 Kompiuterna inzheneriia / Azyrankulov Yehor Anatoliiovych.* – Kyiv, 2023. – 99 s. Rezhym dostupu do resursu: https://ela.kpi.ua/handle/123456789/64438. Data zvernennia:

31.03.2026.
14. Wei Hung Pan, Ming Jie Chok, Jonathan Leong Shan Wong, Yung Xin Shin, Yeong Shian Poon, Zhou Yang, Chun Yong Chong, David Lo, Mei Kuan Lim. "Assessing AI Detectors in Identifying AI-Generated Code: Implications for Education", *arXiv:2401.03676v1 [cs.CE] 8 Jan 2024*. Available: http://arxiv.org/pdf/2401.03676. Accessed on: March 31, 2026.

Історія статті:
Отримано: 11.05.2026 Доопрацьовано: 17.05.2026 Прийнято до друку: 23.05.2026 Опубліковано: 29.05.2026