УДК 378.147:004.4:004.8

Данило Оніщенко
аспірант кафедри інформаційних технологій і програмування
Український державний університет імені Михайла Драгоманова, Київ, Україна
d.s.onischenko@udu.edu.ua
ORCID: 0009-0002-3380-4702

ІНСТРУМЕНТИ ШТУЧНОГО ІНТЕЛЕКТУ В НАВЧАННІ ПРОГРАМУВАННЯ МАЙБУТНІХ УЧИТЕЛІВ ІНФОРМАТИКИ: КРИТЕРІЇ ДОБОРУ ТА ДИДАКТИЧНА СИСТЕМАТИЗАЦІЯ

Анотація. У статті здійснено загальний аналітичний огляд використання інструментів штучного інтелекту в підготовці майбутніх учителів інформатики та сформовано цілісну модель їх педагогічно обґрунтованого добору. Обґрунтовано актуальність переходу від епізодичного застосування генеративних сервісів до системного проєктування інструментально-педагогічного середовища навчання програмування у закладах вищої освіти. На основі аналізу сучасних міжнародних і вітчизняних досліджень охарактеризовано групи інструментів: універсальні моделі великої мови (Large Language Models, LLM), зокрема ChatGPT, Gemini, Claude, Copilot; асистенти кодування, інтегровані в IDE (GitHub Copilot, Codeium, Amazon CodeWhisperer), а також засоби генерації задач, тестів, пояснень, аналізу коду. Запропоновано дидактичну систематизацію за чотирма вимірами: функцією, рівнем втручання, етапами роботи з кодом і типом підтримки. Сформульовано критерії відбору інструментів для методики навчання програмування майбутніх учителів інформатики: дидактична доцільність, функціональна відповідність, етапність застосування, ризики академічної недоброчесності, персоналізація, технічна доступність і професійна придатність для педагогічної практики. Визначено місце процесу їх використання в авторській методиці, у якій інструменти штучного інтелекту розглядаються як засоби підтримки навчальної діяльності, об’єкти методичного аналізу та ресурси формування компетентностей штучного інтелекту майбутнього вчителя. Окремо аргументовано значущість умінь формулювати запити до інструментів штучного інтелекту як складової підготовки з програмування й методичної підготовки. Узагальнено організаційно-педагогічні умови впровадження запропонованої моделі: нормативне регулювання використання інструментів, підготовка викладачів, інституційний моніторинг навчальних результатів і підтримка культури академічної доброчесності. Практична цінність результатів полягає у можливості використання моделі для оновлення змісту дисциплін, проєктування навчальних завдань різних рівнів складності, розроблення прозорих критеріїв оцінювання та планування подальших експериментальних досліджень ефективності інтеграції ШІ у підготовці майбутніх учителів інформатики.

Ключові слова: навчання програмування; майбутні учителі інформатики; генеративний штучний інтелект; LLM-інструменти; асистенти кодування; критерії добору; дидактична систематизація; інженерія запитів

ВСТУП

Стрімка інтеграція генеративних моделей у повсякденні освітні практики зумовлює новий формат навчання програмування у вищій школі. Універсальні LLM-інструменти та спеціалізовані асистенти кодування вже використовуються студентами для пояснення теоретичних понять, пошуку помилок, автодоповнення коду та підготовки навчальних матеріалів (UNESCO, 2023). За таких умов питання полягає не в самому факті використання інструментів штучного інтелекту, а у визначенні меж і педагогічно виправданих сценаріїв їх застосування.

Для спеціальності 014.09 «Середня освіта (Інформатика)» ця проблема має особливу важливість. Здобувач освіти є одночасно суб’єктом опанування програмування та майбутнім учителем, який надалі організовуватиме навчальну діяльність учнів у середовищі, насиченому цифровими інструментами. Отже, підготовка майбутнього вчителя інформатики потребує не лише розвитку власних компетентностей у

програмуванні, а й формування здатності методично доцільно проєктувати використання інструментів штучного інтелекту в шкільному контексті (Шемет, 2025).

Міжнародні політики й аналітичні документи (UNESCO, OECD, професійні спільноти комп'ютерної освіти) акцентують, що впровадження генеративного штучного інтелекту має спиратися на принципи провідної ролі людини, академічної доброчесності, захисту даних і педагогічної доцільності (TeachAI & CSTA, 2025). Для української педагогічної освіти це означає необхідність переходу від реактивної моделі («дозволити/заборонити») до конструктивної моделі («добрати, обмежити, методично вбудувати»).

Аналіз наукових досліджень і публікацій. Огляд джерел засвідчує, що сучасні дослідження застосування інструментів штучного інтелекту у навчанні програмування умовно можна згрупувати в чотири напрями.

Перший напрям – систематичні огляди LLM у комп'ютерній освіті. Вони узагальнюють потенціал інструментів для персоналізованої підтримки, формувального оцінювання, автоматизованого створення навчальних матеріалів та рефлексії; водночас фіксують ризики поверхового засвоєння матеріалу й надмірної залежності студентів від зовнішніх підказок (Raihan et al., 2025).

Другий напрям – емпіричні дослідження результативності LLM-інструментів у курсах програмування. Частина робіт демонструє позитивні результати щодо навчальної мотивації, самоефективності та швидкості виконання завдань, але також вказує на нестійкість якості відповідей у складних задачах та потребу обов'язкової верифікації згенерованих матеріалів (Borgonovi et al., 2025).

Третій напрям – вивчення впливу асистентів кодування (насамперед GitHub Copilot) на виконання завдань з програмування студентами. Дані показують скорочення часу виконання рутинних елементів кодування й зростання продуктивності, але не усувають ризиків некритичного прийняття запропонованих фрагментів без розуміння алгоритмічної логіки.

Четвертий напрям – дослідження формування компетентностей штучного інтелекту майбутніх учителів і методичної інтеграції штучного інтелекту у педагогічну підготовку. У вітчизняному науковому дискурсі наявні праці, присвячені цифровій трансформації освіти, використанню чат-ботів, асистентів та агентів у педагогічній діяльності, а також підготовці майбутніх учителів інформатики до роботи з відповідними технологіями (Умрик & Морзе, 2025).

Водночас порівняльний аналіз показує, що питання *системного добору* інструментів саме для навчання програмування майбутніх учителів інформатики залишається недостатньо опрацьованим. Недостатньо представлені: (1) цілісні класифікації інструментів за педагогічно значущими ознаками; (2) критерії добору, орієнтовані на професійний профіль учителя інформатики; (3) моделі поетапного впровадження інструментів штучного інтелекту у методику навчання програмування

Метою цього дослідження є наукове обґрунтування добору інструментів штучного інтелекту для навчання програмування майбутніх учителів інформатики, їх дидактична систематизація та окреслення місця такого добору в авторській методиці навчання.

Для досягнення мети визначено такі **завдання**: узагальнити сучасні наукові підходи до використання інструментів штучного інтелекту у вищій освіті та навчанні програмування; охарактеризувати основні групи інструментів; систематизувати інструменти за дидактичною функцією, рівнем втручання, етапами роботи з кодом і типом підтримки; обґрунтувати критерії добору інструментів для підготовки майбутнього вчителя інформатики; показати місце добору інструментів у структурі авторської методики навчання програмування; аргументувати значущість умінь

формулювати запити до інструментів штучного інтелекту як компонента професійної компетентності майбутнього педагога.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Дослідження має оглядово-аналітичний характер і ґрунтується на поєднанні методів: теоретичного аналізу джерел, порівняння, класифікації, дидактичної інтерпретації та експертного узагальнення. Емпірична частина в цій статті не передбачалася; отримані результати мають концептуально-методичний статус і призначені для подальшої експериментальної перевірки.

Методика виконання дослідження включала чотири послідовні кроки: відбір доречних джерел за тематичними блоками (LLM у вищій освіті, штучний інтелект у навчанні програмування, підготовка майбутніх учителів до використання інструментів штучного інтелекту, етичні та політичні рамки інтеграції штучного інтелекту); змістовий аналіз функціональних можливостей інструментів та типових сценаріїв їх застосування в навчанні програмування; побудову багатовимірної дидактичної систематизації інструментів і формування критеріїв добору; проєктування місця добору інструментів у логіці авторської методики навчання програмування майбутніх учителів інформатики.

У роботі дотримано принципу розгляду штучного інтелекту *виключно як інструмента*. Аналітичні висновки формулювалися без суб'єктивації інструментів, з фокусом на функціональних можливостях, дидактичних ефектах і педагогічних умовах їх використання.

Для забезпечення внутрішньої узгодженості запропонованої моделі добору використано систему методологічних принципів, які визначають не лише логіку аналізу інструментів, а й логіку їх інтерпретації в контексті підготовки майбутніх учителів інформатики.

Кожен інструмент розглядається через його зв'язок із конкретними результатами навчання, а не через декларативно високі технічні характеристики. У термінах дидактичного проєктування це означає, що інструмент отримує методичний статус лише тоді, коли його використання покращує якість опанування основ програмування, підсилює рефлексивну активність студента або розширює його здатність проєктувати навчальні ситуації для учнів. Отже, модель добору відштовхується від запитання «який тип навчальної діяльності потрібно підсилити?» і лише після цього переходить до запитання «яким інструментом це доцільно реалізувати?» (Ouh et al., 2023).

У моделі закладено положення, що педагогічна доцільність інструмента є функцією етапу підготовки студента. Інструмент із високим генеративним потенціалом може бути продуктивним на етапі проєктної діяльності, але передчасним на етапі формування базових алгоритмічних умінь. Тому добір інструментів здійснюється не як одноразове рішення, а як поетапна конфігурація: від переважно пояснювальних і діагностичних режимів до режимів часткової автоматизації та професійно-орієнтованого використання в педагогічних задачах.

Використання інструмента штучного інтелекту в навчанні програмування є методично припустимим лише за умови можливості перевірки отриманого результату. Верифікація охоплює три рівні: (1) технічний – тестування коду, аналіз граничних випадків, контроль відповідності вимогам задачі; (2) концептуальний – пояснення студентом логіки рішення і причин вибору конкретного підходу; (3) педагогічний – здатність студента трансформувати отримане рішення у формат, придатний для навчання інших (учнів, одногрупників, молодших курсів). Така багаторівнева перевірка дає змогу уникнути зведення навчальної активності до копіювання згенерованих фрагментів. (Zhang et al., 2023)

Добір інструментів має враховувати ризики академічної недоброчесності, питання конфіденційності навчальних даних, ліцензійні обмеження, а також інституційні політики закладу освіти. У зв'язку з цим у моделі передбачається обов'язкова перевірка наявності в інструмента механізмів контролю використання, прозорості джерел і режимів роботи, сумісних з етичними нормами вищої педагогічної освіти.

Інструмент вважається пріоритетним для підготовки майбутнього вчителя інформатики за умови його подвійної корисності: для власного навчання програмування та для майбутньої педагогічної діяльності. Це дозволяє уникнути ситуації, коли студент опановує інструмент, що є технологічно актуальним, але не переноситься в шкільну практику або не підтримує ключові професійні функції вчителя (пояснення, діагностика помилок, формувальне оцінювання, проєктування завдань) (Chugai & Havrylenko, 2024).

Практична реалізація зазначених принципів здійснювалася через побудову аналітичної матриці, у якій кожний клас інструментів оцінювався за низкою дескрипторів: дидактичний потенціал, рівень втручання, тип підтримки, вимоги до контролю, ризик-профіль, професійна придатність. Завдяки цій матриці модель добору набуває прикладного характеру: вона може використовуватися як основа для оновлення робочих програм дисциплін, для конструювання практичних завдань і для розроблення інституційних рекомендацій щодо впровадження інструментів штучного інтелекту.

На підставі проведеного аналітичного огляду визначено три функціонально різні групи інструментів, доречні для навчання програмування майбутніх учителів інформатики.

1) Універсальні LLM-інструменти загального призначення (ChatGPT, Gemini, Claude, Copilot у чат-режимах). Ця група забезпечує діалоговий формат взаємодії, підтримує пояснення теоретичних конструкцій, генерацію прикладів, аналіз помилок, підготовку навчальних матеріалів. Педагогічна цінність групи полягає в тому, що інструменти можуть працювати як засоби пояснювальної, рефлексивної та методичної підтримки. Ризики групи пов'язані з легким доступом до повних рішень типових задач і потребою постійної перевірки коректності відповідей.

2) Асистенти кодування, інтегровані в IDE (GitHub Copilot, Codeium, Amazon CodeWhisperer). Інструменти групи орієнтовані на етапи реалізації коду: контекстне автодоповнення, генерацію фрагментів, часткове вдосконалення коду, підтримку тестування та налагодження. Вони підвищують продуктивність і сприяють наближенню навчальних практик до професійних стандартів галузі. Для педагогічних цілей важливо зберігати контроль за мірою автоматизації, оскільки некритичне прийняття згенерованих фрагментів знижує якість опанування алгоритмічних рішень (The Carpentries Incubator, 2024).

3) Інструменти генерації задач, тестів, пояснень, аналізу коду та налагодження. До групи належать як функції в межах LLM-інструментів, так і API-орієнтовані рішення для автоматизованого створення освітніх матеріалів та попереднього оцінювання. Ця група є особливо значущою для методичної підготовки майбутніх учителів, оскільки безпосередньо підтримує професійні дії педагога: проєктування завдань, формування критеріїв оцінювання, аналіз типових помилок учнів. Обмеження пов'язані з необхідністю експертної валідації кожного згенерованого матеріалу.

Узагальнення отриманих результатів дало змогу подати багатовимірну систематизацію інструментів.

Табл.1
Дидактична систематизація інструментів
штучного інтелекту для навчання програмування

Ознака систематизації	**Категорії та дидактичний зміст**	**Типові інструменти**
За дидактичною функцією	Пояснювальна, генеративна, рефлексивна, оціночна підтримка; функції реалізуються по-різному залежно від навчального сценарію	ChatGPT, Gemini, Claude, Copilot (чат); GitHub Copilot, Codeium, Amazon CodeWhisperer
За рівнем втручання в навчальну діяльність	Низький (локальні підказки), середній (генерація фрагментів із потребою інтеграції студентом), високий (повне рішення з мінімальною участю студента)	Низький/середній: IDE-асистенти, налаштовані чат-режими; високий: неналаштовані генеративні режими
За етапами роботи з кодом	Аналіз умови і планування; написання; налагодження і тестування; документування та пояснення	LLM-інструменти (аналіз, пояснення, документування), асистенти кодування (написання, частково тестування)
За типом підтримки	Пояснення, генерація, перевірка, рефлексія, оцінювання; тип підтримки визначається не лише інструментом, а й способом постановки навчального завдання	Усі класи інструментів за умови дидактичного налаштування

(Джерело: таблицю складено автором самостійно)

Запропонована систематизація має принципове методичне значення: один і той самий інструмент може бути дидактично корисним або шкідливим залежно від ролі, яку йому задає викладач у конкретному виді діяльності. Тому об’єктом педагогічного проєктування є не лише «вибір платформи», а передусім «вибір режиму використання інструмента».

За результатами синтезу проаналізованих підходів сформульовано сім критеріїв добору інструментів для навчання програмування майбутніх учителів інформатики.

1. **Дидактична доцільність.** Інструмент повинен підсилювати навчальну діяльність студента (аналіз, синтез, оцінювання), а не заміщувати її автоматичною генерацією готових рішень (Garcia, 2025).

2. **Функціональні можливості.** Оцінюються режими роботи (пояснення, генерація, перевірка, тестування), стабільність результатів, підтримка мов програмування, що відповідають освітній програмі.

3. **Відповідність етапам навчання програмування.** На початкових етапах пріоритет мають пояснювальні режими та часткові підказки; на старших етапах доцільно розширювати застосування асистентів кодування у проєктних формах роботи (Shihab et al., 2025).

4. **Ризики академічної недоброчесності.** Оцінюються доступність повних рішень, прозорість походження згенерованих матеріалів, можливості контролю використання інструмента та проєктування завдань, стійких до формального копіювання.

5. **Можливість персоналізації.** Важливо, щоб інструмент дозволяв адаптувати складність пояснень, рівень підказок, типи навчальних завдань і формат зворотного зв’язку під потреби конкретного здобувача освіти (Kazemitabaar et al., 2023).

6. **Технічна доступність.** Враховуються вартість, академічні ліцензії, вимоги до інфраструктури та відповідність політикам захисту даних закладу освіти.

7. **Доцільність для майбутнього вчителя інформатики.** Інструмент має підтримувати не лише виконання завдань з програмування студентом, а й педагогічно значущі дії: генерацію дидактичних матеріалів, аналіз типових учнівських помилок,

моделювання оцінювання, проєктування навчальних сценаріїв (Amazon Web Services, 2024-2025).

Для підвищення операційної критеріїв подано узагальнену порівняльну характеристику.

Табл.2

Порівняльна характеристика основних інструментів

Інструмент / клас	**Тип**	**Провідні дидактичні функції**	**Рівень втручання**	**Ключовий ризик**
ChatGPT	Універсальна LLM	Пояснення коду, генерація прикладів і задач, аналіз помилок, підтримка рефлексії	Середній–високий	Доступ до повних рішень без осмислення
Gemini	Універсальна мультимодальна LLM	Пояснення, генерація матеріалів, аналіз великих контекстів	Середній–високий	Потреба налаштування політик доступу й перевірки матеріалів
Claude	Універсальна LLM із великим контекстом	Поглиблені пояснення, аналіз складних фрагментів коду, методичні матеріали	Середній–високий	Ризик надмірної делегації аналітичних дій інструменту
GitHub Copilot	Coding assistant (IDE)	Автодоповнення, генерація фрагментів, підтримка налагодження й відлагодження коду	Середній	Механічне прийняття підказок без перевірки
Codeium	Coding assistant (IDE)	Генерація/відлаголження коду, документування, практичні заняття	Середній	Недостатня критична верифікація пропозицій
Amazon CodeWhisperer	Coding assistant (IDE/AWS)	Автодоповнення, безпековий аналіз, ліцензійні підказки	Середній	Використання як каналу обхідного виконання завдань

(Джерело: таблицю складено автором самостійно)

У межах дисертаційного дослідження добір інструментів розглядається як елемент проєктування цілісного цифрового освітнього середовища, у якому формуються три взаємопов'язані компоненти компетентності майбутнього вчителя інформатики: компонент із програмування, методичний і компонент грамотності у сфері штучного інтелекту.

Перший компонент – підтримка власного навчання програмування. На початковому етапі пріоритет надається пояснювальним сценаріям LLM-інструментів: декомпозиція задач, аналіз повідомлень про помилки, порівняння альтернативних алгоритмів, створення додаткових тренувальних прикладів. На просунутому етапі частка асистентів кодування збільшується в контексті проєктної діяльності та тестування.

Другий компонент – методичний аналіз інструментів. Студенти виконують завдання з оцінювання дидактичних можливостей різних інструментів, моделюють сценарії уроків програмування з урахуванням ризиків академічної недоброчесності, розробляють варіативні задачі, критерії перевірки та схеми надання зворотного зв'язку учням (Kuzu, 2025).

Третій компонент – проєктувальна діяльність. Перспективним є виконання міні проєктів із проєктування локальних або доменно-обмежених освітніх агентів (зокрема на основі підходів із пошуком за джерелами), де майбутній учитель навчається керувати джерельною базою, контролювати якість відповідей і задавати педагогічні обмеження інструмента (Коваль, 2024). Такий формат знижує ризики «чорної скриньки» і підсилює усвідомлене ставлення до інструментів.

Узгоджене застосування трьох компонентів забезпечує перехід від споживацької моделі використання інструментів штучного інтелекту («отримати готовий код») до

професійно-рефлексивної («використати інструмент для навчання, аналізу, методичного проєктування та педагогічного супроводу»).

Узагальнення проаналізованих досліджень засвідчило, що якість навчального ефекту безпосередньо пов'язана з якістю формулювання запитів. За однакової задачі різні формулювання можуть призводити до різних типів відповідей: від поверхового готового фрагмента коду до покрокового пояснення й рефлексивного діалогу (Скворцова & Федерякін, 2025).

Для методики підготовки майбутніх учителів інформатики доцільно виокремлювати щонайменше чотири типи навчальних запитів:

1. *пояснювальні* (орієнтовані на розкриття логіки алгоритму);
2. *діагностичні* (орієнтовані на виявлення помилок і причини некоректної роботи);
3. *методичні* (орієнтовані на створення завдань, критеріїв оцінювання, прикладів учнівських помилок);
4. *рефлексивні* (орієнтовані на порівняння альтернатив і обґрунтування вибору рішення).

Відповідно, підготовка має включати не лише техніку формулювання запитів, а й педагогічні правила їх застосування: заборона запитів, що передбачають пряме виконання контрольних завдань; вимога аргументувати прийняті рішення; обов'язкова верифікація згенерованого коду через тестування та пояснення. Саме така модель взаємодії переводить використання інструментів штучного інтелекту в площину розвитку компетентностей, а не спрощення навчальної діяльності.

Практика інтеграції інструментів штучного інтелекту у навчання програмування засвідчує, що питання ризиків є не периферійним, а системоутворювальним для якості освітнього результату. Саме тому в межах дослідження додатково сформовано матрицю ризиків і запобіжників, яка може використовуватися як робочий інструмент викладача під час планування занять.

Табл. 3

Матриця ризиків використання інструментів штучного інтелекту у навчанні програмування та педагогічних запобіжників

Група ризику	**Типовий прояв у навчальному процесі**	**Дидактичний наслідок**	**Рекомендовані запобіжники**
Надмірна автоматизація	Студент приймає повні згенеровані рішення без самостійного проєктування алгоритму	Зниження рівня продуктивної мисленнєвої діяльності, слабка здатність до перенесення знань на нові задачі	Поетапні обмеження на режими генерації; задачі з обов'язковим поясненням і захистом рішення; оцінювання процесу, а не лише результату
Формальна доброчесність	Робота виглядає «коректною», але не відображає реального внеску студента	Невідповідність між формальною успішністю та фактичною компетентністю	Використання усного захисту, коротких аудиторних мікрозавдань, логів запитів і версійного контролю етапів виконання
Некоректність згенерованих матеріалів	Помилкові пояснення, некоректні тести, хибні припущення щодо складності алгоритму	Формування стійких концептуальних помилок і неадекватних практик перевірки коду	Стандартизований протокол верифікації (тести, контрприклади, взаєморецензування); робота з «навмисно помилковими» прикладами для тренування критичної перевірки
Правові та етичні ризики	Некоректне запозичення коду, порушення ліцензій, неусвідомлена передача чутливих даних	Зниження культури професійної відповідальності майбутнього вчителя	Включення модулів з академічної етики й ліцензування коду; регламент передачі даних до зовнішніх

Група ризику	Типовий прояв у навчальному процесі	Дидактичний наслідок	Рекомендовані запобіжники
			сервісів; аналіз випадків порушень
Технологічна нерівність доступу	Частина студентів не має стабільного доступу до платних або ресурсомістких інструментів	Порушення рівності освітніх можливостей та порівнюваності результатів	Використання академічних або безкоштовних планів; резервні альтернативи; уніфікація обов'язкового інструментарію на рівні курсу
Методична залежність викладача	Викладач переносить на інструмент функції педагогічного проєктування без критичної адаптації	Стандартизація без урахування контексту групи, втрати індивідуалізації навчання	Методичні семінари для викладачів; експертне рецензування згенерованих штучним інтелектом матеріалів; локальні банк-завдань і рубрики оцінювання

(Джерело: таблицю складено автором самостійно)

Використання цієї матриці показує, що ризики мають керований характер і можуть бути інтегровані в дидактичне проєктування курсу не як «підстава для заборони», а як «підстава для педагогічної регуляції». Для підготовки майбутніх учителів інформатики це особливо важливо, оскільки вони мають засвоїти не лише техніку користування інструментами, а й професійну логіку превентивного управління ризиками в учнівському середовищі.

З метою підвищення прикладного потенціалу моделі добору запропоновано сценарний підхід до інтеграції інструментів штучного інтелекту, де кожний етап підготовки майбутнього вчителя інформатики має власну конфігурацію цілей, інструментів і типів контролю.

Табл.4

Сценарна модель інтеграції інструментів штучного інтелекту у підготовці майбутніх учителів інформатики

Етап підготовки	Провідна дидактична мета	Переважні інструменти та режими	Типові навчальні дії студента	Контроль і оцінювання
Початковий (базові курси)	Формування алгоритмічного мислення і базових практик кодування	LLM у пояснювальному/ діагностичному режимі; обмеження повної генерації рішень	Декомпозиція задач, пояснення помилок, побудова псевдокоду, порівняння двох підходів до рішення	Короткі усні захисти, трасування коду вручну, мікротести на розуміння логіки
Середній (прикладне програмування)	Перехід від окремих конструкцій до цілісних програмних рішень	Поєднання LLM і асистентів кодування із контрольованим рівнем втручання	Розробка модулів, тестування, аналіз пропозицій IDE-асистента	Рубрики оцінювання коду, аналіз історії змін, рецензування в парах
Просунутий (проєктна діяльність)	Формування інженерного стилю мислення та відповідальності за якість ПЗ	Асистенти кодування у проєктних задачах; LLM для документації та аналізу архітектури	Проєктування компонентів, оптимізація, підготовка технічної документації, обґрунтування архітектурних рішень	Демонстрація проєкту, взаєморецензування, аналіз дефектів і виправлень

Етап підготовки	Провідна дидактична мета	Переважні інструменти та режими	Типові навчальні дії студента	Контроль і оцінювання
Методичний (курси методики навчання)	Трансформація умінь із програмування у педагогічні дії	LLM для генерації задач, критеріїв оцінювання, типових учнівських помилок; аналіз політик використання штучного інтелекту	Проєктування уроків, створення диференційованих завдань, моделювання формувального зворотного зв'язку	Експертиза конспектів занять, захист методичних рішень, аналіз відповідності етичним вимогам
Педагогічна практика	Апробація інструментів у наближених до шкільних умовах	Обмежений набір інструментів з прозорими правилами використання	Організація навчальної взаємодії учнів з інструментами штучного інтелекту, моніторинг ризиків, корекція сценаріїв	Рефлексивний звіт, спостереження наставника, аналіз навчальних артефактів учнів

(Джерело: таблицю складено автором самостійно)

Запропонована сценарна модель дозволяє уникнути двох крайнощів: повної відмови від інструментів штучного інтелекту та неконтрольованої автоматизації навчальної діяльності. Її цінність полягає в тому, що вона встановлює педагогічно зрозумілу послідовність «ціль – інструмент – діяльність – контроль», яка може бути безпосередньо імплементована в робочі програми дисциплін.

Однією з центральних проблем інтеграції інструментів штучного інтелекту у навчання програмування є перегляд підходів до оцінювання. Традиційна модель, орієнтована переважно на кінцевий результат виконання завдання з програмування, виявляється недостатньою, оскільки не відображає реальну міру самостійності студента, якість його рефлексії та здатність аргументувати прийняті рішення. У зв'язку з цим доцільно переходити до багатокомпонентної системи оцінювання.

У межах запропонованої моделі доцільно виокремлювати чотири групи індикаторів.

Когнітивні індикатори відображають глибину розуміння алгоритмічних концепцій: здатність пояснити часову складність, обґрунтувати вибір структури даних, порівняти альтернативні способи реалізації. Наявність підтримки штучного інтелекту не знімає вимоги до демонстрації такого розуміння, а навпаки посилює її.

Операційні індикатори характеризують якість програмного продукту й процесу його створення: коректність реалізації, покриття тестами, читабельність, стабільність роботи в крайових випадках, якість виправлення дефектів. Тут інструменти штучного інтелекту можуть бути джерелом прискорення, але оцінювання має фіксувати внесок студента в інтеграцію, перевірку та доопрацювання пропозицій.

Рефлексивно-метакогнітивні індикатори виявляють здатність студента аналізувати власну стратегію розв'язання задачі: чому було сформульовано певний запит, чому відхилено або прийнято конкретну пропозицію штучного інтелекту, які помилки виявлено після верифікації. Саме ці індикатори безпосередньо корелюють із формуванням професійної автономності майбутнього вчителя.

Етико-професійні індикатори фіксують рівень академічної доброчесності, коректність роботи з джерелами коду, дотримання політик конфіденційності та здатність формулювати правила безпечного використання штучного інтелекту для учнів.

Для забезпечення валідності оцінювання доцільно комбінувати кілька форматів: автоматизовані тести, аудиторний захист, рецензування в малих групах, короткі

рефлексивні звіти, а також аналіз робочих артефактів (версії коду, історія змін, шаблони запитів). У такій моделі оцінюється не лише *що* зроблено, а й *як* і *чому* це зроблено.

Практичний досвід використання такого підходу показує, що прозора система критеріїв знижує мотивацію до формального копіювання готових рішень, оскільки студент розуміє: підсумкова оцінка визначається не «наявністю коду», а сукупністю доказів його компетентності. Для підготовки майбутніх учителів інформатики це має додатковий ефект - вони отримують готову модель оцінювання, яку згодом можуть адаптувати до шкільних умов.

Ефективність запропонованої моделі добору визначається не лише якістю її теоретичного обґрунтування, а й наявністю інституційних умов для впровадження. Аналіз джерел і узагальнення практик дозволяють визначити щонайменше п’ять таких умов.

1) Нормативно-методична умова. На рівні кафедри або освітньої програми мають бути закріплені прозорі правила використання інструментів штучного інтелекту: дозволені режими, обмеження під час контрольних заходів, вимоги до декларування використання інструментів, протоколи верифікації результатів. Відсутність таких правил породжує неоднозначність і знижує довіру до оцінювання.

2) Дидактико-проєктувальна умова. Робочі програми дисциплін повинні містити явно прописані сценарії застосування інструментів відповідно до цілей теми та рівня підготовки студентів. У межах цієї умови важливо формувати банк завдань, стійких до тривіальної генерації повного розв’язку, а також пакет шаблонів запитів для різних видів навчальної активності.

3) Кадрово-компетентнісна умова. Викладачам потрібна системна підтримка у формуванні методичної компетентностей штучного інтелекту: семінари, взаємне рецензування курсів, обмін практиками оцінювання, робота з випадками етичних і правових ризиків. Без цієї умови технологічні новації можуть залишатися на рівні фрагментарних ініціатив.

4) Технологічно-інфраструктурна умова. Заклад освіти має забезпечити мінімально достатній і рівнодоступний інструментарій: підтримувані IDE, доступні акаунти, інструктивні матеріали, регламент зберігання даних. Важливим є принцип «педагогічно необхідного мінімуму»: студент не повинен опинятися в нерівних умовах через відсутність доступу до конкретного платного сервісу.

5) Моніторингова умова. Впровадження моделі потребує регулярного збору даних: типові помилки студентів, динаміка навчальних результатів, частота використання різних режимів інструментів, випадки порушення доброчесності, запити викладачів на методичну підтримку. Саме моніторинг дає змогу коригувати модель добору та зберігати її актуальність.

Сукупність зазначених умов формує інституційний контур, у якому інструменти штучного інтелекту стають не ситуативним додатком, а структурованим компонентом підготовки майбутнього вчителя інформатики.

У контексті підготовки майбутніх учителів інформатики доцільно говорити не лише про «техніку формулювання запитів», а про *культуру запитів* — інтегрований комплекс когнітивних, методичних і етичних норм взаємодії з інструментами штучного інтелекту.

Культура запитів включає три взаємопов’язані компоненти:

Когнітивний компонент передбачає чітке структурування запиту: формулювання задачі, зазначення обмежень, опис вхідних умов, уточнення очікуваного формату відповіді. Така структура мінімізує неоднозначність і підвищує якість пояснювальної підтримки.

Методичний компонент визначає дидактичну мету запиту. Для майбутнього вчителя принципово важливо вміти переводити запит із площини «дай готовий код» у площину «поясни, постав навідні запитання, запропонуй варіанти завдань різної складності, сформуй критерії оцінювання». Саме такий зсув забезпечує педагогічну цінність інструмента.

Етичний компонент включає заборону на несанкціоноване отримання готових відповідей для контрольних робіт, коректне декларування використання інструментів штучного інтелекту, дотримання норм академічної доброчесності та критичне ставлення до отриманих матеріалів.

Для формування культури запитів у межах курсів програмування можуть використовуватися спеціальні вправи: порівняння якості відповідей на «слабкі» і «сильні» запити; доопрацювання запитів із метою переходу від генеративного до пояснювального режиму; аналіз «помилкових» запитів, що призводять до некоректних рішень; розроблення педагогічних запитів для шкільних навчальних ситуацій. Такий формат підготовки формує у студентів досвід свідомого керування інструментом, а не пасивного споживання його відповідей.

У підсумку культура запитів виступає однією з ключових умов перетворення інструментів штучного інтелекту на дидактично керовані засоби навчання програмування, сумісні з вимогами професійної підготовки майбутнього вчителя інформатики.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Узагальнення сучасних міжнародних і вітчизняних досліджень підтверджує доцільність використання інструментів штучного інтелекту в навчанні програмування у вищій школі за умови їх педагогічно керованої інтеграції. Для підготовки майбутніх учителів інформатики запропоновано багатовимірну систематизацію інструментів за дидактичною функцією, рівнем втручання, етапами роботи з кодом і типом підтримки, що забезпечує операційну основу для проєктування навчальних сценаріїв. Обґрунтовано систему критеріїв добору інструментів (дидактична доцільність, функціональність, етапність, ризики недоброчесності, персоналізація, технічна доступність, професійна придатність), яка дозволяє ухвалювати методично виважені рішення щодо їх включення в освітній процес. Показано, що в авторській методиці навчання програмування добір інструментів виконує не допоміжну, а системоутворювальну функцію: інструменти одночасно підтримують власне навчання студента, слугують об’єктами методичного аналізу та формують його готовність до педагогічного супроводу учнів у насиченому інструментами штучного інтелекту середовищі. Доведено, що уміння формулювати педагогічно доцільні запити до інструментів штучного інтелекту є суттєвою складовою професійної підготовки майбутнього вчителя інформатики, оскільки саме якість запиту визначає тип отримуваної підтримки і її навчальну цінність.

Практичне значення отриманих результатів полягає в тому, що запропонована модель може бути безпосередньо використана для оновлення змісту дисциплін «Програмування», «Методика навчання інформатики», «Програмування робототехнічних систем», а також для розроблення локальних положень щодо використання інструментів штучного інтелекту у навчальному процесі. На рівні освітньої програми модель забезпечує прозоре узгодження між результатами навчання, типами завдань, інструментальними засобами й критеріями оцінювання. На рівні викладача вона виконує функцію навігаційної карти, що дозволяє обрати інструмент не за принципом технологічної новизни, а за принципом педагогічної доцільності. На рівні студента модель задає зрозумілу логіку відповідального використання підтримки

штучного інтелекту: від пояснення й діагностики до професійно орієнтованого проєктування навчальних матеріалів.

Крім того, результати дослідження можуть бути покладені в основу внутрішнього забезпечення якості освіти через уніфікацію мінімальних вимог до верифікації згенерованого коду, стандартизацію рубрик оцінювання робіт, виконаних із використанням інструментів штучного інтелекту, та розроблення модулів підвищення кваліфікації викладачів щодо дидактично доцільної інтеграції штучного інтелекту в курси програмування.

Перспективи подальших досліджень пов'язані з емпіричною перевіркою запропонованої моделі добору інструментів у межах експериментального навчання, оцінюванням впливу різних конфігурацій інструментів на компетентності студентів у програмуванні, методичну компетентність та компетентності у сфері штучного інтелекту, а також розробленням валідних діагностичних інструментів для вимірювання сформованості навичок педагогічно орієнтованої взаємодії з сервісами штучного інтелекту.

Конфлікт інтересів

Автор заявляє про відсутність потенційного конфлікту інтересів фінансового, професійного, наукового чи особистого характеру, що міг би вплинути на результати дослідження, їх інтерпретацію, рецензування або ухвалення редакційних рішень.

Декларування використання інструментів штучного інтелекту

Під час підготовки дослідження автор використовував Perplexity.ai для пошуку релевантних наукових джерел та їх подальшого аналітичного опрацювання. Після використання інструмента матеріали були додатково перевірені, інтерпретовані та відредаговані автором. Повну відповідальність за зміст і достовірність поданої публікації несе автор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

UNESCO (2023). Guidance for generative AI in education and research. Paris, France: UNESCO. https://doi.org/10.54675/EWZM9535

Шемет, Д. (2025). Підготовка майбутніх учителів інформатики до інтеграції штучного інтелекту у навчальний процес на прикладі чат-бота. *Електронне наукове фахове видання "Відкрите освітнє е-середовище сучасного університету", 19*, 255-271. https://doi.org/10.28925/2414-0325.2025.1917

TeachAI, & CSTA. (2025, July 7). Guidance on the Future of Computer Science Education in an Age of AI. TeachAI. https://www.teachai.org/cs-old-july

Raihan, N., Siddiq, M., Santos, J., & Zampieri, M. (2025). Large language models in computer science education: A systematic literature review. arXiv. https://doi.org/10.48550/arXiv.2410.16349

Borgonovi, F., Bastagli, F., Ochojska, M., & Piumatti, G. (2025). AI adoption in the education system: International insights and policy considerations for Italy. *OECD Artificial Intelligence Papers, 52*, 1-100. https://doi.org/10.1787/69bd0a4a-en

Умрик, М. & Морзе, Н. (2025). Використання ботів, асистентів, агентів штучного інтелекту в освітній діяльності. *Електронне наукове фахове видання "Відкрите освітнє е-середовище сучасного університету", 19*, 205-225. https://doi.org/10.28925/2414-0325.2025.1914

Ouh, E., Gan, B., Shim, K., & Wlodkowski, S. (2023). ChatGPT, can you generate solutions for my coding exercises? An evaluation on its effectiveness in an undergraduate Java programming course. CoRR, abs/2305.13680. https://doi.org/10.48550/arXiv.2305.13680

Zhang, B., Liang, P., Zhou, X., Ahmad, A., & Waseem, M. (2023). Practices and challenges of using GitHub Copilot: An empirical study. CoRR, abs/2303.08733. https://doi.org/10.48550/arXiv.2303.08733

Chugai, O., & Havrylenko, K. (2024). ChatGPT: Attitudes and experiences of technical university students in Ukraine. *Information Technologies and Learning Tools*, *101*(3), 15-27. https://doi.org/10.33407/itlt.v101i3.5559

The Carpentries Incubator (2024). AI-assisted Coding with Codeium. GitHub Copilot. https://carpentries-incubator.github.io/gen-ai-coding/

Garcia, M. (2025). Teaching and learning computer programming using ChatGPT: A rapid review of literature amid the rise of generative AI technologies. *Education and Information Technologies, 30*, 16721-16745. https://doi.org/10.1007/s10639-025-13452-5

Shihab, M., Hundhausen, C., Tariq, A., Haque, S., Qiao, Y., Wise, B., & Sanchez, C. (2025). The effects of GitHub Copilot on computing students' programming effectiveness, efficiency, and processes in brownfield coding tasks. In Proceedings of the 2025 ACM Conference on International Computing Education Research (ICER 2025). ACM. https://doi.org/10.1145/3702652.3744219

Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In Proceedings of the 23rd Koli Calling International Conference on Computing Education Research (pp. 1-12). ACM. https://doi.org/10.1145/3631802.3631806

Amazon Web Services (2024-2025). Amazon Q Developer / Amazon CodeWhisperer documentation. https://docs.aws.amazon.com/amazonq/latest/qdeveloper-ug/service-rename.html

Kuzu, S. (2025). Artificial intelligence in pre-service teacher education: Bibliometrics analysis. *Pedagogical Research, 10*(4), em0249. https://doi.org/10.29333/pr/17402

Скворцова, С., & Федерякін, Д. (2025). Промпт-інжиніринг як сучасна компетентність педагога. *Наукові записки. Серія: Педагогічні науки.*

Надходження статті до видання 15.03.2026 р.
Прийняття статті до друку після рецензування 15.04.2026 р.
Дата публікації 24.04.2026 р.

ARTIFICIAL INTELLIGENCE TOOLS IN PROGRAMMING EDUCATION FOR PRE-SERVICE COMPUTER SCIENCE TEACHERS: SELECTION CRITERIA AND DIDACTIC SYSTEMATIZATION

Danylo Onishchenko
PhD student
Department of Information Technology and Programming
Dragomanov Ukrainian State University, Kyiv, Ukraine
d.s.onischenko@udu.edu.ua
ORCID: 0009-0002-3380-4702

Abstract. This article provides a comprehensive analytical review of artificial intelligence tools used in programming education for pre-service computer science teachers and proposes a didactically grounded model for their pedagogical selection and integration. The study addresses a

practical and methodological challenge faced by teacher education programs: the transition from occasional and unsystematic use of generative AI services to an intentionally designed instructional environment in which tools are selected according to learning outcomes, stage of student development, and academic integrity requirements. The research synthesizes international policy documents, systematic reviews, and empirical studies on large language models and coding assistants, including ChatGPT, Gemini, Claude, GitHub Copilot, Codeium, Amazon Q Developer, and related instructional applications. Based on comparative analysis and didactic interpretation, the paper introduces a multidimensional systematization of AI tools by educational function, level of intervention, phase of work with code, and type of support. It further substantiates criteria for tool selection in the context of preparing future teachers: didactic relevance, functional fit, stage-appropriate use, verifiability of generated outputs, ethical and legal compatibility, accessibility, and dual professional utility for both personal learning and future classroom practice. The paper also argues that prompt formulation competence should be treated as a core learning outcome in programming methodology courses, because the quality of prompts directly affects explanation depth, student reflection, and the risk profile of tool use. In addition, the article defines organizational and pedagogical conditions required for sustainable implementation, including institutional regulations, teacher professional development, infrastructure support, and monitoring procedures. The proposed model contributes a practical framework for curriculum design and for quality assurance of AI-supported learning activities, and it establishes a methodological basis for future experimental validation in higher pedagogical education. The findings also specify implementation conditions for higher education institutions, including policy alignment, teacher professional development, transparent assessment rubrics, and continuous monitoring of student learning outcomes, which strengthens the practical applicability of the proposed model in curriculum renewal.

Keywords: programming education; pre-service computer science teachers; generative artificial intelligence; LLM tools; coding assistants; selection criteria; didactic systematization; prompt engineering

REFERENCES (TRANSLATED AND TRANSLITERATED)

UNESCO (2023). Guidance for generative AI in education and research. Paris, France: UNESCO. https://doi.org/10.54675/EWZM9535

Shemet, D. (2025). Preparing future computer science teachers for integrating artificial intelligence into the educational process: The case of a chatbot. *Electronic Scientific Professional Journal "Open educational e-environment of modern university", 19*, 255-271. https://doi.org/10.28925/2414-0325.2025.1917 (in Ukrainian).

TeachAI, & CSTA (2025, July 7). Guidance on the Future of Computer Science Education in an Age of AI. TeachAI. https://www.teachai.org/cs-old-july

Raihan, N., Siddiq, M., Santos, J., & Zampieri, M. (2025). Large language models in computer science education: A systematic literature review. arXiv. https://doi.org/10.48550/arXiv.2410.16349

Borgonovi, F., Bastagli, F., Ochojska, M., & Piumatti, G. (2025). AI adoption in the education system: International insights and policy considerations for Italy. *OECD Artificial Intelligence Papers, 52*, 1-100. https://doi.org/10.1787/69bd0a4a-en

Umryk, M., & Morze, N. (2025). Using bots, assistants, and artificial intelligence agents in educational activities. *Electronic Scientific Professional Journal "Open educational e-environment of modern university", 19*, 205-225. https://doi.org/10.28925/2414-0325.2025.1914 (in Ukrainian).

Ouh, E., Gan, B., Shim, K., & Wlodkowski, S. (2023). ChatGPT, can you generate solutions for my coding exercises? An evaluation on its effectiveness in an undergraduate Java programming course. CoRR, abs/2305.13680. https://doi.org/10.48550/arXiv.2305.13680

Zhang, B., Liang, P., Zhou, X., Ahmad, A., & Waseem, M. (2023). Practices and challenges of using GitHub Copilot: An empirical study. CoRR, abs/2303.08733. https://doi.org/10.48550/arXiv.2303.08733

Chugai, O., & Havrylenko, K. (2024). ChatGPT: Attitudes and experiences of technical university students in Ukraine. *Information Technologies and Learning Tools, 101*(3), 15-27. https://doi.org/10.33407/itlt.v101i3.5559

The Carpentries Incubator (2024). AI-assisted Coding with Codeium / GitHub Copilot. https://carpentries-incubator.github.io/gen-ai-coding/

Garcia, M. (2025). Teaching and learning computer programming using ChatGPT: A rapid review of literature amid the rise of generative AI technologies. *Education and Information Technologies, 30*, 16721-16745. https://doi.org/10.1007/s10639-025-13452-5

Shihab, M., Hundhausen, C., Tariq, A., Haque, S., Qiao, Y., Wise, B., & Sanchez, C. (2025). The effects of GitHub Copilot on computing students' programming effectiveness, efficiency, and processes in brownfield coding tasks. In Proceedings of the 2025 ACM Conference on International Computing Education Research (ICER 2025). ACM. https://doi.org/10.1145/3702652.3744219

Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In Proceedings of the 23rd Koli Calling International Conference on Computing Education Research (pp. 1-12). ACM. https://doi.org/10.1145/3631802.3631806

Amazon Web Services (2024-2025). Amazon Q Developer / Amazon CodeWhisperer documentation. https://docs.aws.amazon.com/amazonq/latest/qdeveloper-ug/service-rename.html

Kuzu, S. (2025). Artificial intelligence in pre-service teacher education: Bibliometrics analysis. *Pedagogical Research, 10*(4), em0249. https://doi.org/10.29333/pr/17402

Koval, O. (2024). Using artificial intelligence in programming education in blended learning settings. *Electronic Scientific Professional Journal "Open educational e-environment of modern university", 17,* 65-78. https://doi.org/10.28925/2414-0325.2024.175 (in Ukrainian).

Skvortsova, S., & Federiakin, D. (2025). Prompt engineering as a contemporary competence of an educator. *Naukovi zapysky. Seriia: Pedahohichni nauky* (in Ukrainian).