Володимир МАРЧУК,
здобувач другого (магістерського) рівня вищої освіти
Бердянського державного педагогічного університету
(м. Запоріжжя)
e-mail: volodymyr.marchuk3004@gmail.com

Науковий керівник:
Микола КУДІНОВ,
кандидат педагогічних наук,
доцент кафедри фізики, математики та методики навчання
Бердянського державного педагогічного університету
(м. Запоріжжя)

ВІД СПОЖИВАЧА ЦИФРОВОГО КОНТЕНТУ ДО ЙОГО ТВОРЦЯ: РОЛЬ STEM-ГУРТКА У РОЗВИТКУ ЦИФРОВИХ КОМПЕТЕНТНОСТЕЙ ШКОЛЯРІВ

Учень постійно має справу з цифровими технологіями, але між умінням користуватися готовим сервісом і здатністю створити власну програму є відчутна різниця. Саме програмування дає можливість побачити цифрове середовище з іншого боку: не лише отримати готовий результат, а зрозуміти, як він формується. Для учнів 7-10 класів STEM-гурток може стати місцем, де такий перехід відбувається поступово, без вимоги мати попередній досвід. Навчання починається з основ Python, а далі прості вправи переходять у невеликі програми, які можна змінювати та доповнювати. Додаткової уваги ця тема потребує через поширення генеративного штучного інтелекту. Отримати готовий код сьогодні нескладно, тому цінність навчання дедалі більше пов’язана з розумінням алгоритму, умінням перевірити результат, знайти помилку і пояснити власне рішення. У такій ситуації STEM-гурток дає школяреві простір не тільки для знайомства з технологіями, а й для їх осмисленого використання та створення власних цифрових продуктів.

Розвиток цифрової компетентності є одним із помітних напрямів сучасної шкільної освіти. У Концепції Нової української школи цифрові вміння розглядаються в контексті компетентнісного навчання, де важливе місце займають самостійність учня та практичне застосування знань [1]. Державний стандарт базової середньої освіти також визначає

інформаційно-комунікаційну компетентність серед ключових і пов'язує її з роботою з інформацією та цифровими технологіями [2]. Для STEM-освіти практична складова має особливе значення. Концепція її розвитку та методичні рекомендації передбачають дослідницькі, проєктні й практико-орієнтовані завдання [3; 4]. Подібний підхід простежується і в європейській рамці DigComp 2.2, де створення цифрового контенту виділено як окрему сферу цифрової компетентності [5]. Останніми роками до цієї теми додалося питання штучного інтелекту. UNESCO звертає увагу не лише на можливості таких систем, а й на необхідність критично оцінювати їхні відповіді, розуміти обмеження та зберігати активну роль людини [6; 7]. В українських рекомендаціях щодо використання ШІ у школі також наголошено на відповідальному та усвідомленому застосуванні цих технологій [8].

Метою дослідження є розгляд можливостей STEM-гуртка з програмувальним спрямуванням для розвитку цифрової компетентності учнів 7–10 класів, а також визначення місця програмування та інструментів штучного інтелекту в такій роботі. У процесі підготовки тез використано аналіз нормативних і методичних джерел, спостереження за виконанням учнями практичних завдань та узагальнення досвіду організації гурткових занять.

Починати навчання програмування доцільно з невеликих задач, у яких результат зрозумілий учневі ще до написання коду. Під час знайомства зі змінними, типами даних, введенням та виведенням інформації такою роботою може бути простий калькулятор. Користувач вводить два числа, програма виконує дію і показує результат. Поруч із калькулятором можна запропонувати обчислення площі та периметра, конвертер температури або переведення одиниць вимірювання. У цих програмах небагато коду, тому увага зосереджується на логіці дій, а не на великій кількості нових конструкцій.

Після знайомства з умовами з'являються задачі, у яких програма має реагувати на різні дані. Це може бути визначення парного числа, порівняння кількох значень або вибір математичної операції в калькуляторі. Тут природно виникає потреба в тестуванні. Програма, яка правильно спрацювала один раз, може дати інший результат після введення нуля, від'ємного числа чи неочікуваного значення. Так учень поступово звикає

перевіряти код, а не обмежуватися першим успішним запуском. Цикли дозволяють перейти до задач, які вже більше нагадують завершені невеликі програми. Простий приклад – таблиця множення. Більш ігровий варіант – «Вгадай число», де комп'ютер випадково обирає число, а користувач отримує підказки після кожної спроби. Базову версію легко розширити: додати кількість спроб, рівні складності або можливість почати гру ще раз. Подібно можна побудувати вікторину з кількома запитаннями та підрахунком балів. Пізніше до неї додаються списки запитань, випадковий порядок або різні підсумкові повідомлення. Такі завдання зручні ще й тим, що допускають різний рівень складності. Не обов'язково вимагати однакового результату від учнів 7 і 10 класу або від тих, хто працює з різною швидкістю. Для одного учня достатньо базового алгоритму, для іншого наступним кроком стане перевірка введення, функції, збереження результату чи додаткова логіка. Спільна тема при цьому зберігається. Важливим є повернення до вже написаного коду. Після вивчення функцій можна переробити стару програму і поділити її на логічні частини, після знайомства зі списками змінити спосіб зберігання даних. Така робота показує прогрес значно наочніше, ніж окремі вправи, які після виконання більше не відкриваються. Частина учнів продовжує експериментувати з програмами вдома, додаючи власні зміни або намагаючись знайти інший спосіб розв'язання. Генеративний ШІ в цій системі не варто розглядати ні як заборонений інструмент, ні як заміну самостійній роботі. Його користь залежить від того, яке завдання ставиться перед учнем. Якщо просто скопіювати умову і отримати готову програму, навчальна цінність буде невеликою. Інша ситуація виникає, коли учень уже має власну спробу і звертається до ШІ, щоб зрозуміти повідомлення про помилку, підібрати тестові дані або розібрати незнайому конструкцію. Відповідь після цього все одно потрібно перевірити в середовищі програмування [6, с. 18–19; 8]. Код, створений ШІ, може бути й окремим навчальним матеріалом. Його можна прочитати, знайти умови та цикли, перевірити на різних значеннях, а потім змінити правила програми. Не менш корисно порівняти два варіанти розв'язання однієї задачі. У такій роботі головним стає не походження коду, а здатність учня зрозуміти його і аргументувати зміни. Робота із запитами до ШІ теж має зв'язок з алгоритмізацією. Фраза «напиши калькулятор на Python» майже нічого не говорить про вимоги до програми. Потрібно

уточнити, які операції вона підтримує, які дані отримує, як реагує на помилки та що має показати користувачеві. Фактично це повертає учня до формулювання задачі та її умов. Рамка компетентностей UNESCO з ШІ для учнів також поєднує технічні навички з критичним мисленням та відповідальним використанням технологій [7]. Під час перевірки невеликих програм корисно звертати увагу не лише на кінцевий файл. Учень може коротко пояснити алгоритм, показати кілька тестів і змінити одну з умов. Якщо використовувався ШІ, достатньо пояснити, на якому етапі він був потрібний і як перевірялася отримана відповідь. Такий формат дозволяє оцінити не тільки те, чи запускається програма, а й наскільки учень розуміє власну роботу.

Отже, STEM-гурток з програмувальним спрямуванням створює умови для поступового розвитку цифрової компетентності: від перших команд Python до невеликих програм, які учень може перевіряти, змінювати та розширювати. Практичні завдання дають можливість поєднати вивчення синтаксису з алгоритмічним мисленням, пошуком помилок і самостійним прийняттям рішень. Різний рівень складності одного завдання дозволяє працювати з учнями 7–10 класів без вимоги однакового результату для всіх. Генеративний ШІ може доповнювати таку роботу, якщо використовується для пояснення, аналізу й перевірки, а не для механічної заміни власного рішення. У цьому випадку програмування допомагає школяреві перейти від звичного користування цифровими технологіями до розуміння їхньої логіки та створення власних цифрових продуктів.

Список використаних джерел:

1. Нова українська школа: концептуальні засади реформування середньої школи / за заг. ред. М. Грищенка. Київ, 2016. 40 с.

2. Про деякі питання державних стандартів повної загальної середньої освіти : постанова Кабінету Міністрів України від 30.09.2020 р. № 898. URL: https://zakon.rada.gov.ua/laws/show/898-2020-%D0%BF#Text (дата звернення: 08.10.2026).

3. Про схвалення Концепції розвитку природничо-математичної освіти (STEM-освіти) : розпорядження Кабінету Міністрів України від 05.08.2020 № 960-р. URL: https://zakon.rada.gov.ua/laws/show/960-2020-%D1%80#Text (дата звернення: 08.10.2026).

4. Методичні рекомендації щодо розвитку STEM-освіти в закладах загальної середньої та позашкільної освіти у 2025/2026 навчальному році : лист ІМЗО від 18.07.2025 № 21/08-624. URL: https://imzo.gov.ua/2025/08/08/lyst-imzo-vid-18-07-2025-21-08-624-metodychni-rekomendatsii-shchodo-rozvytku-stem-osvity-v-zakladakh-zahal-noi-seredn-oi-ta-pozashkil-noi-osvity-u-2025-2026-navchal-nomu-rotsi/ (дата звернення: 08.10.2026).

5. Vuorikari R., Kluzer S., Punie Y. DigComp 2.2: The Digital Competence Framework for Citizens. Luxembourg : Publications Office of the European Union, 2022. DOI: 10.2760/115376.

6. Miao F., Holmes W. Guidance for Generative AI in Education and Research. Paris : UNESCO, 2023. 44 p.

7. Miao F., Shiohira K., Lao N. AI Competency Framework for Students. Paris : UNESCO, 2024. 80 p. DOI: 10.54675/JKJB9835.

8. Інструктивно-методичні рекомендації щодо запровадження та використання технологій штучного інтелекту в закладах загальної середньої освіти / Міністерство освіти і науки України, Міністерство цифрової трансформації України. 2024. 62 с. URL: https://mon.gov.ua/static-objects/mon/sites/1/news/2024/05/21/Instruktyvno.metodychni.rekomendatsiyi.shchodo.SHI.v.ZZSO-22.05.2024.pdf (дата звернення: 08.10.2026).