МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

МЕТОДИКА ПІДГОТОВКИ УЧНІВ 10-11(12) КЛАСІВ ДО УЧАСТІ В ОЛІМПІАДАХ З ПРОГРАМУВАННЯ

Кваліфікаційна робота студентки
групи Ім-24
ступінь вищої освіти магістр
спеціальності
014.09 Середня освіта (Інформатика)
Желудько Яни Вікторівни

Керівник: к. пед. н., доцент,
Шокалюк С. В.

Кривий Ріг – 2025

ЗАПЕВНЕННЯ

Я, Желудько Яна Вікторівна, розумію і підтримую політику Криворізького державного педагогічного університету з академічної доброчесності. Запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Я не надавала і не одержувала недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають покликання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Криворізького державного педагогічного університету ознайомлена. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

ЗМІСТ

ВСТУП

У сучасному світі, де інформаційні технології є рушієм інновацій та розвитку суспільства, підготовка висококваліфікованих ІТ-фахівців стає стратегічним завданням. Олімпіади з програмування є одним із найефективніших інструментів для раннього виявлення та розвитку талановитої молоді, здатної до алгоритмічного мислення та вирішення нетривіальних завдань.

Підготовка до олімпіад з програмування учнів 10-11(12) класів вимагає особливих методичних підходів. Вона виходить далеко за межі стандартної шкільної програми, охоплюючи складні розділи дискретної математики, теорії графів, динамічного програмування та структур даних. Ефективність такої підготовки напряму залежить від індивідуального підходу та кваліфікованого тьюторського супроводу.

Наукова новизна роботи полягає у теоретичному обґрунтуванні та розробці моделі тьюторського супроводу індивідуалізованої підготовки учнів до олімпіад з програмування. Методику можна буде використати у шкільних гуртках, факультативах, STEM-центрах або системі позашкільної освіти.

Актуальність дослідження зумовлена низкою визначальних чинників:

– стрімким зростанням ролі ІТ-галузі та стратегічним значенням підготовки висококваліфікованих інженерних кадрів;

– недостатністю стандартної шкільної програми з інформатики для формування глибоких алгоритмічних знань, необхідних для участі у змаганнях;

– потребою в індивідуалізованих моделях навчання, що враховують різний темп засвоєння матеріалу та стартовий рівень учнів;

– критичною роллю вчителя-ментора у наданні своєчасного, персоналізованого зворотного зв’язку, який не можуть забезпечити виключно автоматизовані системи перевірки.

Мета дослідження полягає у теоретичному обґрунтуванні, розробці та експериментальній перевірці ефективності методики індивідуалізованої

підготовки учнів 10-11(12) класів до участі в олімпіадах з програмування в умовах тьюторського супроводу.

Об'єкт дослідження: навчання олімпіадного програмування учнів старших класів.

Предмет дослідження: зміст, форми, методи індивідуалізованої підготовки учнів 10-11(12) класів до олімпіад з програмування.

Завдання дослідження:

1. Узагальнити та систематизувати теоретичні засади підготовки учнів старших класів до олімпіад з програмування в науково-педагогічній літературі.
2. Дослідити психолого-педагогічні особливості та дидактичні принципи індивідуалізованого навчання алгоритмізації.
3. Обґрунтувати використання платформи Eolymp як засобу індивідуалізації підготовки учнів.
4. Здійснити аналіз змісту та типології завдань ІІ (районного) етапу Всеукраїнської учнівської олімпіади з інформатики (2022–2025 рр.)
5. Розробити навчальну програму спецкурсу «Основи олімпіадного програмування».
6. Скласти методичні рекомендації для вчителів щодо організації інтенсивної підготовки учнів-початківців.

Методи дослідження:

– аналіз науково-методичної літератури з проблем методики навчання інформатики та олімпіадного програмування;

– систематизація та узагальнення передового педагогічного досвіду підготовки учнів до змагань;

– вивчення та узагальнення функціоналу інформаційно-освітньої платформи E-Olymp для індивідуальної роботи;

– теоретичне обґрунтування та створення методики індивідуалізованої підготовки.

Практичне значення роботи полягає в:

– розробці навчальної програми курсу за вибором «Основи олімпіадного програмування» (27 годин), зміст якої сформовано на основі аналізу завдань ІІ етапу Всеукраїнської учнівської олімпіади з інформатики (2022–2025 рр.);

– створенні методичних рекомендацій для вчителів інформатики щодо організації тренувальних змагань у середовищі Eolymp за моделлю «Прискорене вивчення – Аналіз – Дорішування»;

– формуванні структурованого банку різнорівневих задач та матеріалів для аналізу типових помилок, що дозволяє реалізувати індивідуальну траєкторію підготовки учнів 10-11(12) класів.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ЗАСАДИ ПРОБЛЕМИ ПІДГОТОВКИ УЧНІВ ДО ОЛІМПІАД З ПРОГРАМУВАННЯ

1.1. Олімпіади з програмування як ключовий інструмент розвитку алгоритмічного мислення

У сучасному інформаційному суспільстві, де цифрові технології та програмування є фундаментальними, олімпіади з програмування (також відомі як «змагальне програмування«) відіграють виняткову роль в освітній системі. Вони є не лише формою інтелектуальних змагань, але й визнаним потужним освітнім інструментом та значним мотиваційним стимулом для учнів та студентів [10]. Дослідження підтверджують, що участь у змаганнях з програмування робить значний внесок в освіту у сфері обчислювального мислення та навичок вирішення комплексних проблем [11]. Таким чином, підготовка до таких змагань стає важливою та невід'ємною складовою сучасної ІТ-освіти [1].

Центральною метою олімпіадного руху є розвиток алгоритмічного мислення. Це поняття розглядається багатьма науковцями як фундаментальна компетентність, що лежить в основі успішного вивчення програмування. За твердженням Опанасюк (2024) *«Алгоритмічне мислення – це здатність формулювати послідовність дій для розв'язання задачі, оптимізуючи її структуру й ресурси»*[5]. Авторка прямо вказує на тісний зв'язок між процесом вивчення програмування та формуванням специфічних розумових операцій, що складають алгоритмічне мислення. З когнітивної точки зору, ці розумові операції є мисленнєвими структурами, або «схемами», до них ми можемо віднести аналіз, прогнозування та узагальнення. Процес навчання, особливо під час розв'язання складних завдань, спрямований на формування та автоматизацію цих схем. Як зазначає Джон Свеллер (1988) у своїй роботі про когнітивне навантаження, саме побудова таких схем, що зберігають знання про певний клас проблем та способи їх вирішення, є ключовим ефектом навчання, який відрізняє експертне мислення від мислення новачків [6]. Інші дослідники

також активно вивчають та пропонують різні шляхи та методики для ефективного розвитку цього мислення в учнів [7]. Олімпіадні задачі, завдяки своїй складності та необхідності пошуку оптимальних і нетривіальних розв'язків, виступають у цьому процесі найпотужнішим каталізатором.

Окрім когнітивного розвитку, олімпіади з програмування слугують ключовим інструментом для виявлення, підтримки та роботи з обдарованими учнями. Ефективна робота з цією категорією школярів потребує особливих умов, зокрема впровадження інноваційних підходів та активного використання інтерактивних технологій. Як зазначає Дем’янчук (2025), підтримка обдарованих учнів у сучасному освітньому середовищі може ефективно реалізовуватися через міжпредметну взаємодію, особливо на перетині технологій та інформатики [8].

Сучасна підготовка до олімпіад неможлива без відповідної технологічної інфраструктури. Платформи та системи автоматизованої перевірки завдань (Codeforces, E-Olymp, Olympkh) є невід'ємною частиною процесу, забезпечуючи учням можливість миттєво тестувати свої рішення та отримувати об'єктивний зворотний зв'язок [3]. Різноманітні інтерактивні платформи активно використовуються школярами 6–11 класів для вивчення програмування, що підтверджує їхню високу ефективність та поширеність [9]. Стійкість олімпіадного руху була доведена і в кризових умовах: зокрема, у період пандемії COVID-19 проведення шкільних олімпіад з інформатики було успішно адаптоване до дистанційних форматів саме завдяки таким технологіям [4].

Таким чином, необхідність у розробці спеціалізованих методичних засад підготовки учнів посилюється через значний розрив між вимогами стандартної шкільної програми, наприклад, у рамках НУШ [10], та високим рівнем складності олімпіадних завдань. Для досягнення вагомих результатів необхідна інтеграція інноваційних підходів [2], що виходять за межі традиційного уроку та спрямовані на глибоке, часто індивідуалізоване, засвоєння складних алгоритмічних тем.

Можна стверджувати, що олімпіади з програмування є комплексним

освітнім явищем. Саме атмосфера змагання спонукає до пошуку оптимальних рішень і розвитку наполегливості, що є невід'ємними рисами алгоритмічного мислення. Платформи та системи автоматизованої перевірки створюють середовище для постійної практики, миттєвого отримання зворотного зв'язку та самостійного вдосконалення. Це дозволяє учням швидко коригувати свої помилки, аналізувати різні підходи та формувати навички ефективного мислення в умовах обмежених ресурсів і часу. Дистанційна форма освіти сьогодення довела свою ефективність у кризових умовах, забезпечивши безперервність навчання та доступність участі для широкого кола учнів. Вони сприяють розвитку самостійності, дисципліни та здатності працювати в цифровому середовищі.

1.2. Аналіз стану проблеми у науково-методичній літературі

Проблема пошуку та впровадження ефективних методик підготовки учнів до олімпіад з програмування знаходиться в центрі уваги багатьох вітчизняних та зарубіжних науковців, а також педагогів-практиків. Аналіз фахової літератури дозволяє виділити декілька ключових, хоча й взаємопов'язаних, напрямів, в яких ведеться ця робота (Таблиця 1.1).

Особливо актуальним є аналіз сучасного стану олімпіадного руху безпосередньо в Україні. Офіційною основою для цього руху є нормативні документи, такі як накази Міністерства освіти і науки [25]. Дослідники приділяють значну увагу аналізу специфіки проведення ключових етапів Всеукраїнських учнівських олімпіад, а також відстежують та аналізують результати участі українських школярів у міжнародних змаганнях [19]. Важливим є той факт, що педагогічна спільнота активно реагує на виклики сьогодення [15], зокрема, досліджуючи досвід та адаптуючи методики проведення олімпіад в умовах пандемії чи інших кризових ситуацій, та обговорюючи загальні питання підтримки олімпіадного руху в країні [20].

Домінуючою тенденцією у сучасній методиці підготовки є перехід до використання цифрових інструментів та інноваційних технологій. Дослідження

підтверджують, що впровадження інноваційних технологій [24] та моделей, зокрема STEM-освіти [18], позитивно впливає на якість підготовки. Цей процес значно прискорився через вимушений перехід до дистанційних форм роботи, що стимулювало розвиток методик дистанційної підготовки учнів [23].

Таблиця 1.1

Ключові напрями дослідження проблеми підготовки до олімпіад

Напрям	**Ключові аспекти**
Організаційний (переважно в Україні)	Специфіка проведення етапів (I,II, III тури); адаптація до кризових умов (пандемія, воєнний стан); аналіз участі у міжнародних змаганнях.
Технологічний та інструментальний	Використання інтернет-ресурсів та онлайн-платформ; методики дистанційної підготовки; застосування онлайн-змагань у навчанні; розробка інструментів (наприклад, класифікаторів задач)
Змістовно-методичний	Розвиток логічного мислення; впровадження STEM-моделей; розробка навчальних планів; вплив на мотивацію учнів; методики переходу від внутрішніх до міжнародних змагань.

У цьому контексті ключову роль відіграють онлайн-платформи та інтернет-ресурси. Вони розглядаються як ефективний інструмент підготовки [13], що дозволяє впроваджувати методики навчання через онлайн-платформи [28] та використовувати формат онлайн-змагань у системі практичної підготовки [15]. Важливою складовою таких платформ є системи автоматизованої перевірки, які, зокрема, можуть використовуватися для класифікації та добору задач для новачків [27].

З точки зору змісту та методичних підходів, науковці пропонують розробку специфічних навчальних планів, заснованих на змагальному підході [17], що описують шлях від внутрішніх змагань до міжнародних [29]. Окремим та фундаментально важливим напрямом є розвиток логічного мислення учнів старших класів при навчанні програмування [22], для чого розробляються

спеціалізовані логічні задачі та системи інформаційної підтримки вивчення логіки [21]. Ефективність таких підходів також підтверджується позитивним впливом олімпіад на загальну мотивацію учнів до вивчення інформатики [17].

Більш глибокий аналіз джерел дозволяє виявити певні відмінності у фокусі вітчизняних та зарубіжних досліджень. Вітчизняні науковці, що цілком природно, значною мірою зосереджені на *організаційно-адаптивних аспектах*: як провести олімпіади в умовах реформування освіти та зовнішніх кризових викликів, як підтримати національний олімпіадний рух та проаналізувати виступи саме українських команд. Натомість, представлені зарубіжні дослідження є більш *глобально-методичними* або *технічними*: вони пропонують загальні моделі побудови навчальних планів, аналізують загальний шлях до міжнародних змагань або розробляють конкретні технічні інструменти, наприклад, для автоматичної класифікації задач.

Таким чином, аналіз науково-методичної літератури та передового досвіду засвідчує, що основна увага дослідників (як в Україні, так і за кордоном) зосереджена на організаційних аспектах (включно з дистанційними), впровадженні онлайн-платформ та розвитку логічного мислення. Водночас, питання розробки цілісної методики, яка б фокусувалася саме на індивідуалізованій підготовці та ролі тьюторського супроводу з наданням персоналізованих коментарів, залишається недостатньо розкритим, що і визначає предмет даного дослідження.

1.3. Психолого-педагогічні особливості та дидактичні принципи індивідуалізованого навчання учнів 10-11(12) класів

Підготовка учнів 10–11(12) класів до олімпіад з програмування має ґрунтуватися не лише на предметному змісті, але й на глибокому розумінні психолого-педагогічних особливостей цієї вікової групи. Старший шкільний вік є періодом активного професійного самовизначення [37], формування психосоціального здоров'я та розвитку пізнавальної самостійності, особливо в процесі дослідницької діяльності [38]. У контексті олімпіадної підготовки, яка

переважно залучає обдарованих учнів [8], інформатична освітня галузь має значний потенціал для діагностики та розвитку цієї обдарованості [31]. Саме ці особливості – вища мотивація, орієнтація на майбутню професію та здатність до самостійної роботи – роблять традиційні уніфіковані підходи недостатньо ефективними.

Це висуває на перший план сучасні дидактичні принципи, що ставлять в центр освітнього процесу учня як активного, свідомого суб'єкта [36]. Ключовими серед них є диференціація, індивідуалізація та персоналізація навчання.

Принцип диференціації навчання, тобто врахування індивідуальних відмінностей учнів, є фундаментальним для профільної старшої школи. Сучасні дослідження в галузі викладання програмування підкреслюють важливість диференційованого підходу, наприклад, через постійний моніторинг прогресу, автоматизоване формування груп та інші методики.

Тісно пов'язаний принцип індивідуалізації навчання набуває особливої актуальності в умовах інформатизації освіти [40] та є ключовим у роботі з учнями профільної школи, зокрема в умовах змішаного навчання. На відміну від диференціації (адаптації до особливостей групи учнів), індивідуалізація передбачає розробку освітнього маршруту, адаптованого до потреб, здібностей та темпу конкретного учня.

Логічним розвитком цих ідей, особливо в епоху цифрових технологій, є персоналізація навчання. Цей підхід, що є вкрай важливим для підготовки майбутніх ІТ-фахівців, активно використовує цифрові інструменти [41] та адаптивні навчальні платформи [35] для створення унікального освітнього досвіду.

Практичною реалізацією цих принципів є проєктування та формування ***індивідуальної освітньої траєкторії (ІОТ)*** [34]. Сучасні інформаційно-комунікаційні інструменти та спеціалізовані інформаційні системи [33] дозволяють ефективно будувати ІОТ для старшокласників, враховуючи їхні освітні запити. Такі підходи доведені як для формальної [32],

так і неформальної освіти фахівців комп'ютерного профілю [36], дозволяючи створювати цифрові персоналізовані навчальні програми [42], що є критично важливим для підготовки до олімпіад.

На основі аналізу наукових джерел можна стверджувати, що індивідуалізація та персоналізація навчання є не лише технологічними чи організаційними принципами, а передусім психолого-педагогічними умовами ефективного розвитку алгоритмічного мислення.

Авторське бачення ***індивідуалізації*** полягає у розумінні її як цілеспрямованого врахування пізнавальних, інтелектуальних і мотиваційних особливостей кожного учня при доборі змісту, темпу та форм роботи. Індивідуалізація забезпечує оптимальне співвідношення між рівнем складності завдань і когнітивними можливостями учня, створюючи умови для поступового розвитку його мислення.

Персоналізація, у свою чергу, означає надання старшокласнику більшої автономії у побудові власної освітньої траєкторії, виборі напрямів і типів задач, що відповідають його інтересам та рівню готовності. У такій моделі педагог виконує роль тьютора-фасилітатора освітнього процесу, який забезпечує зворотний зв’язок і підтримує рефлексію учня.

Психологічні особливості старших школярів – домінування аналітичного мислення, високий рівень когнітивної активності, потреба у визнанні та самореалізації – безпосередньо впливають на методику підготовки до олімпіад. Саме тому ефективні стратегії мають поєднувати когнітивний і особистісно-орієнтований підходи: розвиток мисленнєвих структур через складні задачі, формування рефлексивних умінь через аналіз власних рішень та підтримку внутрішньої мотивації через ситуації успіху і командну взаємодію.

1.4. Роль тьюторського супроводу та персоналізованого зворотного зв’язку в освітньому процесі

Як було обґрунтовано в попередньому підпункті, ефективна підготовка учнів до олімпіад з програмування вимагає переходу до індивідуалізованих та

персоналізованих моделей навчання. Специфіка олімпіадної підготовки – робота з обдарованими учнями, високе когнітивне навантаження та необхідність розв’язання нетривіальних, нестандартних задач – робить групові та суто технологічні підходи недостатньо ефективними. Сама по собі технологічна реалізація індивідуальної освітньої траєкторії не є достатньою. Ключова роль у цій системі відводиться вчителю, чия функція трансформується з традиційного лектора на педагога-ментора або тьютора. Саме тьюторський супровід є тим механізмом, що дозволяє реалізувати справжню індивідуалізацію.

Дослідження підтверджують ефективність тьюторської діяльності у навчанні інформатики [51] та її фундаментальне значення в освітньому процесі [52]. Особливо це стосується роботи зі здібними студентами в галузі комп’ютерних наук та підготовки до інтелектуальних змагань. Робота з обдарованими учнями має бути системною та вимагає спеціальних методик і технологій педагогічного супроводу.

Авторське бачення ролі тьютора в олімпіадній підготовці полягає в тому, що він виступає не просто як «пояснювач» складних тем, а як діагност та архітектор навчального процесу. На відміну від автоматизованих систем, тьютор здатен діагностувати не просто помилку в коді, а концептуальну прогалину в мисленнєвих структурах (схемах) учня. Він будує індивідуальну траєкторію, вирішуючи, коли учень готовий перейти від жадібних алгоритмів до динамічного програмування, і забезпечує психологічну підтримку, необхідну для подолання фрустрації при вирішенні складних завдань, що є типовим для змагань.

Ключовим інструментом тьютора є персоналізований зворотний зв’язок. На відміну від автоматизованих систем, які зазвичай надають бінарну оцінку («правильно/неправильно») або стандартизовані технічні коментарі, тьютор пропонує глибокий якісний аналіз. Попри швидкий розвиток технологій автоматизованої перевірки програмного коду [49], вони переважно охоплюють лише синтаксичні й технічні аспекти. Діагностика помилкових уявлень,

інтерпретація логіки учня та формування мисленнєвих структур наразі залишаються сферою тьюторської роботи. Такий підхід органічно узгоджується з концепцією формувального оцінювання [44].

Оксаною Демидович та Аліною Карапетян було розглянуто Agile-педагогіку [45] як гнучкий та адаптивний підхід до організації навчання, який переносить принципи Agile-розробки програмного забезпечення в освітній процес. Він базується на коротких навчальних ітераціях, постійному зворотному зв'язку, співпраці та готовності змінювати навчальну траєкторію відповідно до потреб учня. Така модель орієнтується не на статичну програму, а на динамічне коригування цілей, рівня складності та змісту завдань відповідно до індивідуальних особливостей здобувача освіти.

У контексті підготовки учнів старших класів до олімпіад з програмування Agile-педагогіка є концептуально сумісною з принципами персоналізації та тьюторського супроводу. Ітеративність дозволяє будувати підготовку через серію мікроциклів (розв'язання задач → аналіз → корекція стратегії), а постійний фідбек допомагає вчасно виявляти прогалини у мисленні, формувати стійкі алгоритмічні структури та поступово ускладнювати траєкторію розвитку. Гнучкість Agile відповідає потребам обдарованих учнів, для яких стандартний темп навчання є недостатнім та які потребують швидкої адаптації навчального контенту під індивідуальний прогрес.

Тьюторський фідбек в олімпіадній підготовці можна умовно класифікувати на кілька видів:

- ***Реактивний (коригувальний):*** відповідь на пряме запитання учня (*Чому мій алгоритм отримує Time Limit?*);
- ***Проактивний («Scaffolding«):*** випереджувальна підтримка, коли тьютор бачить, що учень «застряг«, і надає мінімально необхідну підказку (натяк на структуру даних або помилкову логіку), не даючи готового рішення;
- ***Концептуальний***: аналіз не лише конкретної помилки, а й загального підходу учня (*Ти використав перебір, але ця задача ефективно розв'язується методом «розділяй і володарюй«*);

- ***Метакогнітивний:*** фідбек, спрямований на рефлексію учня над власними стратегіями навчання та розв’язання задач [48] (*Твій код працює, але він нечитабельний. Як можна було б його вдосконалити для кращого розуміння?*).

У міжнародній педагогічній практиці проактивний підхід описується терміном «scaffolding» (педагогічна підтримка, «будівельні риштування»), який був вперше введений Вудом, Брунером та Россом (1976) [53]. Мета-аналіз підтверджує позитивний вплив «скаффолдингу» на результати навчання програмуванню у школярів [54]. Дослідження доводять, що метакогнітивний «скаффолдинг» позитивно впливає на мотивацію та здатність учнів до перенесення знань, а також допомагає керувати когнітивним навантаженням при вирішенні складних завдань.

На практиці, персоналізований зворотний зв'язок від тьютора найчастіше реалізовується через «рев'ю коду» (code review). Хоча цей метод часто досліджується в контексті взаємної перевірки (peer review), його цінність є значно вищою при асиметричній взаємодії «учень-ментор». Сучасні технології активно розвиваються у напрямку посилення цієї взаємодії, наприклад, через створення інтелектуальних тьюторських систем, використання ШІ для персоналізації або розробку чат-ботів для надання персоналізованого зворотного зв'язку [50].

Таким чином, огляд науково-методичної літератури підтверджує, що майбутнє ефективної підготовки до олімпіад лежить у синергії індивідуальних освітніх траєкторій, реалізованих на онлайн-платформах з адаптивним тестуванням [47], та глибокого педагогічного супроводу з боку вчителя-тьютора, який надає своєчасний, розгорнутий та багаторівневий персоналізований зворотний зв'язок.

Висновки до розділу 1

У першому розділі здійснено комплексний теоретичний аналіз проблеми підготовки учнів старших класів до олімпіад з програмування, що дозволило сформулювати наступні висновки.

Олімпіади з програмування є ключовим інструментом розвитку алгоритмічного мислення та формування фахових компетентностей майбутніх ІТ-спеціалістів. Вони виконують не лише змагальну, але й важливу освітню функцію, стимулюючи учнів до пошуку оптимальних рішень, аналізу та прогнозування, що сприяє формуванню стійких когнітивних структур. Олімпіадні задачі, завдяки своїй нестандартності, виступають каталізатором розвитку обдарованості та пізнавальної самостійності учнів 10–11 класів.

Аналіз вітчизняної та зарубіжної літератури засвідчив, що увага науковців зосереджена переважно на трьох напрямах: *організаційно-адаптивному* (проведення змагань у кризових умовах), *технологічному* (використання онлайн-платформ) та *змістовно-методичному* (розвиток логічного мислення). Водночас питання методики індивідуалізованої підготовки новачків з використанням автоматизованих систем перевірки та моделі тьюторського супроводу залишається недостатньо розкритим, що актуалізує тему дослідження.

Ефективність підготовки старшокласників залежить від врахування їхніх вікових особливостей: професійного самовизначення, потреби у самореалізації та здатності до рефлексії . Це зумовлює необхідність переходу від уніфікованих методик до принципів індивідуалізації (адаптації до темпу учня) та персоналізації (автономії у виборі траєкторії) навчання. Реалізація цих принципів можлива через побудову ***індивідуальних освітніх траєкторій*** з використанням сучасних цифрових інструментів.

Встановлено, що технологічна складова (платформи, системи тестувань) є необхідною, але недостатньою умовою успіху. Вирішальну роль відіграє вчитель-тьютор, який трансформує свою функцію з транслятора знань на

архітектора навчального процесу . Тьютор забезпечує багаторівневий зворотний зв'язок (реактивний, проактивний, концептуальний), здійснює діагностику помилкових концепцій та надає підтримку у зоні найближчого розвитку учня, чого наразі не можуть повноцінно забезпечити автоматизовані системи.

Отже, теоретичний аналіз підтверджує, що ефективна модель підготовки до олімпіад має базуватися на синергії сучасних хмарних сервісів (для автоматизації рутинної перевірки) та методично обґрунтованого тьюторського супроводу (для формування стратегічного мислення). Це створює підґрунтя для розробки методичних засад індивідуалізованої підготовки, чому присвячено другий розділ роботи.

РОЗДІЛ 2. МЕТОДИЧНІ ЗАСАДИ ІНДИВІДУАЛІЗОВАНОЇ ПІДГОТОВКИ ДО ОЛІМПІАД З ПРОГРАМУВАННЯ

2.1. Обґрунтування вибору хмарного сервісу Eolymp як засобу індивідуалізації навчання

В умовах цифровізації освіти та зростання популярності олімпіадного руху, традиційні методи перевірки програмного коду вчителем вручну стають неефективними. Вони не забезпечують миттєвого зворотного зв'язку, який є критично важливим для формування алгоритмічного мислення. Для реалізації розробленої методики підготовки учнів 10-11(12) класів нами було обрано хмарну платформу Eolymp (https://eolymp.com/uk), яка є стандартом проведення змагань з інформатики в Україні.

Вибір саме цього середовища зумовлений низкою технічних та методичних факторів, які дозволяють реалізувати принцип індивідуалізації навчання.

По-перше, ключовою перевагою Eolymp є система автоматичного тестування рішень. Учень відправляє код, і система миттєво перевіряє його на заздалегідь підготовленому наборі вхідних даних (тестах). Це забезпечує об'єктивність — оцінка не залежить від суб'єктивного погляду вчителя, а базується виключно на коректності роботи алгоритму та його ефективності (часу виконання та використання пам'яті).

Для учня це створює умови «швидкого зворотного зв’язку»: він одразу бачить вердикт системи (AC — Accepted, WA — Wrong Answer, TLE — Time Limit Exceeded) і може самостійно шукати помилку, не чекаючи наступного уроку (рис. 2.1).

По-друге, дана платформа використовується для проведення всіх етапів Всеукраїнської учнівської олімпіади з інформатики, про що зазначено на офіційному сайті олімпіади UOI (https://www.uoi.ua/). Доступ до архіву задач минулих років дозволяє вчителю формувати тренувальні набори, які повністю відповідають рівню реальних змагань. У розробленій нами програмі буде

використано задачі саме з цього архіву (рис. 2.2), що гарантує релевантність підготовки.

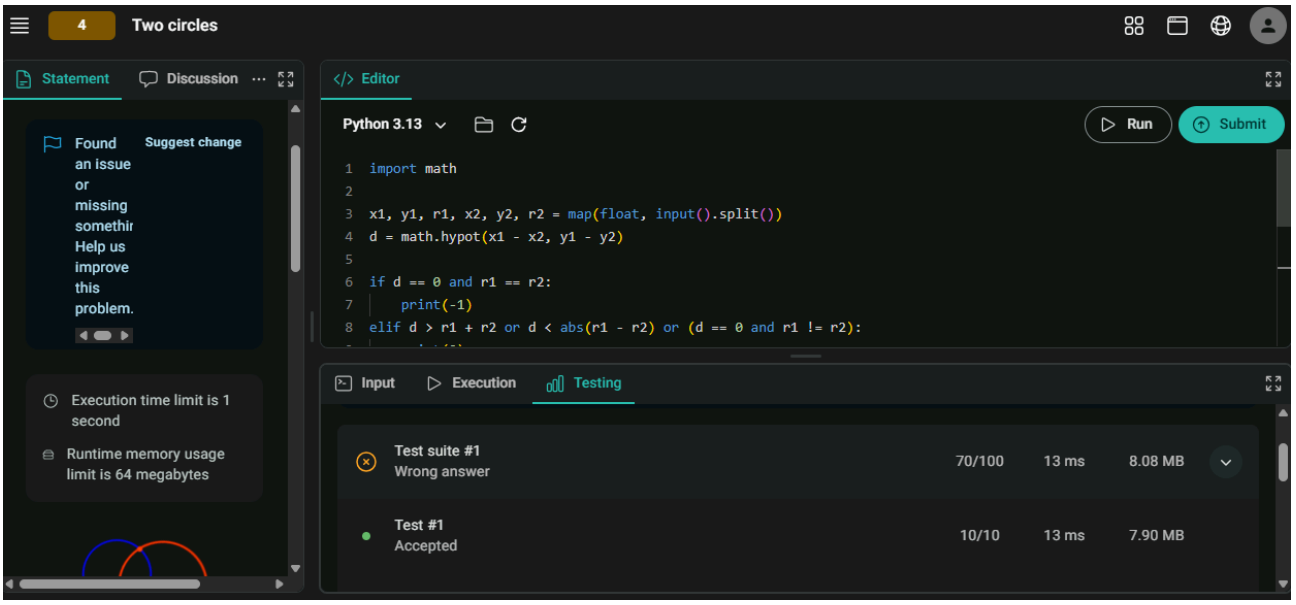

Рис. 2.1. Інтерфейс відправки розв'язку задачі та отримання вердикту системи на платформі Eolymp

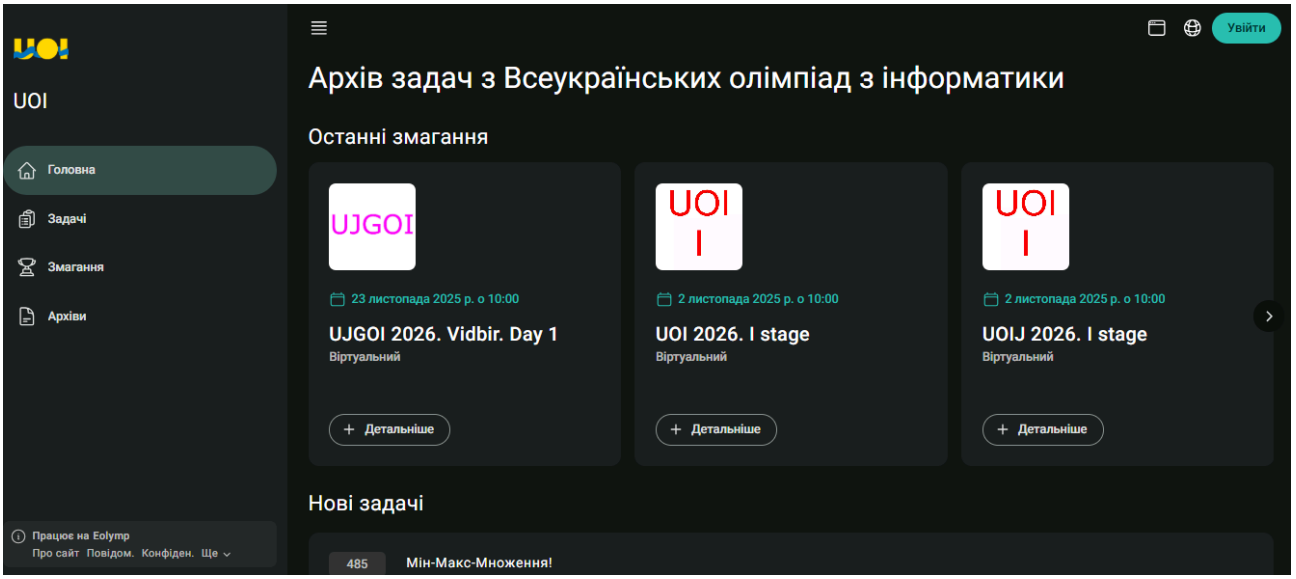

Рис. 2.2. Архів задач UOI на платформі Eolymp

По-третя, функціонал платформи дозволяє вчителю створювати власні «Змагання» (Contests) і гнучко налаштовувати правила їх проведення, що є основою нашої методики «Прискорене вивчення – Аналіз – Дорішування«.

В ***режимі змагання (Contest)*** вчитель обмежує час виконання (наприклад, 60 хвилин). Це створює психологічні умови, наближені до олімпіадних, і вчить учнів тайм-менеджменту.

В ***режимі дорішування (Upsolving)*** після завершення основного часу учні можуть продовжувати здавати задачі. Це ключовий елемент індивідуалізації: учні з повільним темпом засвоєння матеріалу мають можливість розібратися з задачею без стресу.

Крім того, слід виділити можливість вибору системи оцінювання: IOI (часткові бали за тести) або ICPC (все або нічого). Для початківців ми рекомендуємо використовувати режим IOI, оскільки він мотивує учнів, нараховуючи бали навіть за неповні або неоптимальні розв’язки.

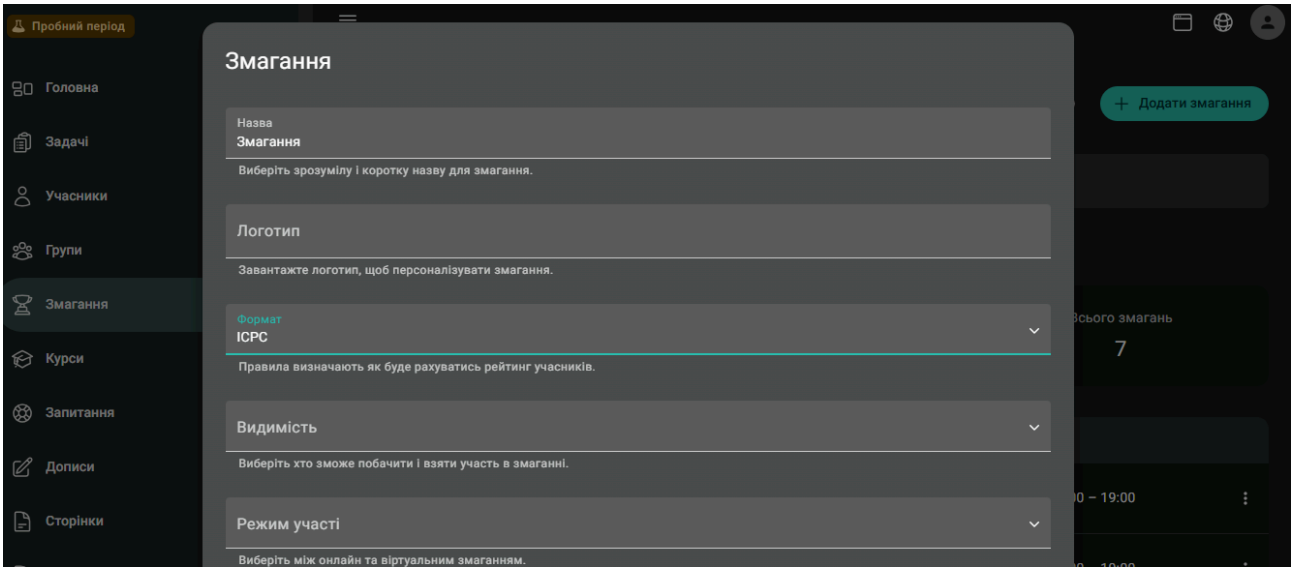

Рис. 2.3. Налаштування параметрів змагання: вибір часу, формату оцінювання IOI/ICPC та доступу

Платформа дозволяє створити власний віртуальний «Простір» (Space). Простір являє собою ізольоване цифрове середовище (загальний обліковий запис), у межах якого адміністратор (вчитель) має можливість публікувати авторські задачі, створювати змагання, вести архіви та керувати списком учасників (організувати учнів у «Групи). Це дає вчителю інструменти для детальної аналітики. У режимі реального часу педагог бачить загальну таблицю результатів, що дозволяє миттєво виявляти учнів, які потребують допомоги

(тих, хто «завис» на одній задачі), і тих, хто потребує додаткового навантаження. Автоматизація перевірки рутинних задач вивільняє час вчителя на уроці. Замість перевірки синтаксису десяти учнівських програм, вчитель виконує роль ментора: проводить code-review (аналіз учнівського коду) після перевірки, пояснює складні алгоритми та надає індивідуальні консультації у разі труднощів.

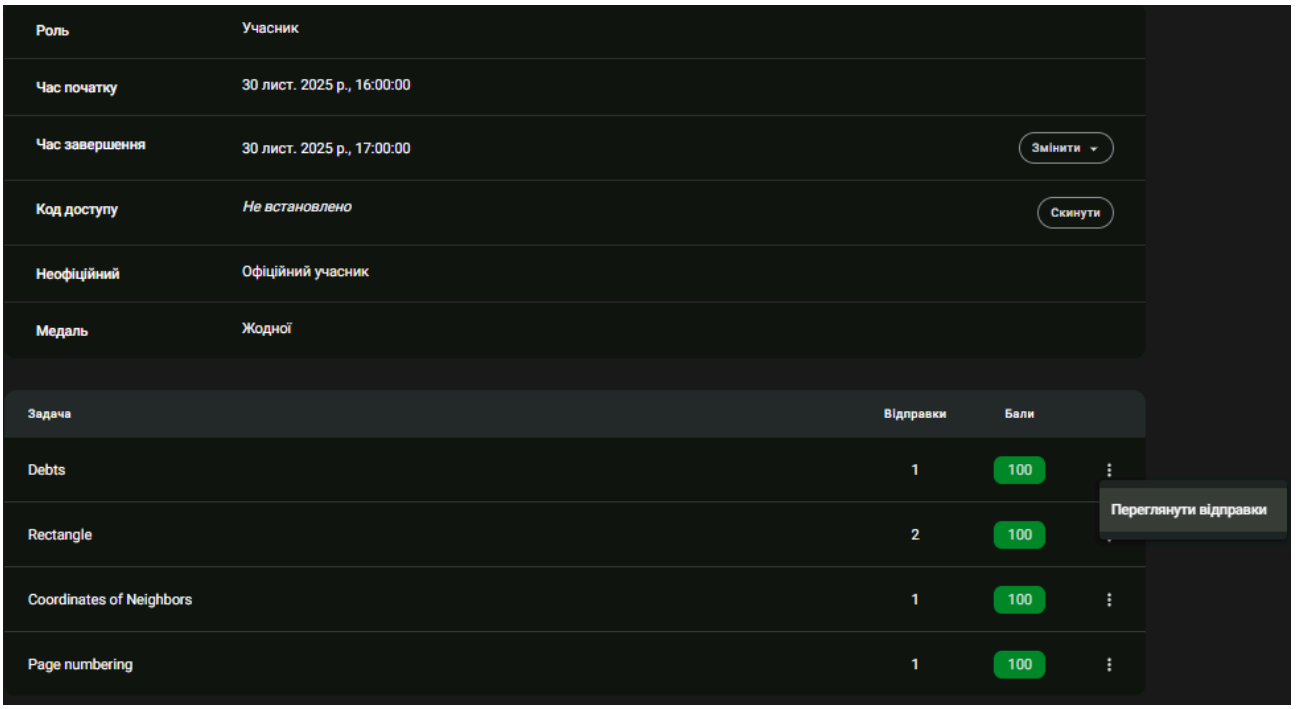

Рис. 2.4. Таблиця розв'язків задач певного учня у реальному часі з детальним аналізом

Eolymp має багатомовний інтерфейс, який дозволяє залучити учасників-закордонників. Також підтримує безліч мов програмування. Проте для нашого курсу обрано Python. У поєднанні з автоматизованою перевіркою це дає ефект «швидкого старту«. Лаконічний синтаксис Python дозволяє учням 10-11(12) класів зосередитися на логіці алгоритму та математичній моделі задачі, а не на технічних деталях коду. Платформа коректно обробляє специфіку Python (наприклад, довгу арифметику), що спрощує розв'язання багатьох задач початкового рівня (тип A та B).

Слід зазначити, що розробники сайту Eolymp змінили свою політику і ввели обмеження для безкоштовних акаунтів, щоб монетизувати платформу [55]. Кожен новостворений простір отримує унікальну веб-адресу на піддомені

eolymp.space, яка генерується автоматично, але підлягає персоналізації в налаштуваннях. Залежно від потреб навчального закладу, простори класифікуються на два типи:

Публічні простори: забезпечують відкритий доступ до матеріалів (задач, дописів, змагань) для будь-яких користувачів, включно з анонімними.

Приватні простори: обмежують доступ до контенту. При переході за посиланням користувач потрапляє на форму авторизації й отримує доступ до даних лише після успішного входу та наявності відповідних прав. Реєстрація учасників у просторі здійснюється адміністратором вручну або шляхом активації опції самостійної реєстрації, що дозволяє учням долучатися до простору автоматично.

Платформа пропонує гнучку ієрархічну систему контролю доступу, що базується на Політиці. Вона визначає набір дозволених дій для конкретного користувача стосовно певного ресурсу (простору, задачі, змагання чи курсу), зокрема:

- Якщо до одного ресурсу застосовано декілька політик, їхні дозволи сумуються.
- Базовими діями є «Читання» (перегляд ресурсу) та «Запис» (внесення змін). Специфічні ресурси мають розширені дії, наприклад, «Тестування» для задач (керування тестами та перевірками) або «Призначення» для курсів (керування студентами).
- Для забезпечення безперервного функціонування простору система вимагає наявності принаймні однієї політики з повним доступом («Дозволити всі дії») для одного з користувачів.

Використання розширеного функціоналу просторів регулюється тарифними планами, які передбачають щомісячну або щорічну оплату. Вартість формується на основі кількості «місць». Поняття «місце» визначає кількість унікальних користувачів, які можуть здійснити вхід у простір протягом розрахункового місяця. Якщо ліміт місць вичерпано, нові користувачі не зможуть увійти до системи до розширення тарифного плану або початку нового

циклу оплати. Важливо зазначити, що видалення учасника звільняє місце для іншого користувача. Платформа надає пробний період терміном 30 днів для ознайомлення з повним функціоналом, що майже ідеально підходить для цілей нашої роботи.

Незважаючи на широкі можливості, безкоштовна версія (Free Plan) або пробні версії мають суттєві обмеження, які можуть впливати на організацію навчального процесу у великих класах. Основні обмеження наведено у Таблиці 2. 1.

Таблиця 2.1

Обмеження функціоналу створення Простору (Space) на платформі Eolymp (Базовий / Безкоштовний тариф)

Параметр / Функціонал	**Характеристика обмеження**	**Вплив на навчальний процес**
Кількість просторів	**1**	Унеможливлює створення окремих ізольованих середовищ для різних класів або груп (наприклад, окремо для 10-А та 10-Б). Усі учні змушені перебувати в одній загальній групі.
Кількість учасників	**До 10 учнів**	Є критичним обмеженням для шкільного застосування. Групи з наповнюваністю понад 10 осіб не можуть бути додані до одного простору для централізованого моніторингу без переходу на платний тариф.
Архів задач	**Безлімітний**	Доступ до загального банку задач не обмежується. Це дозволяє організовувати роботу без використання Простору (через прямі посилання), однак без

		функцій автоматизованого збору статистики групи.
Автоматична перевірка	**Включено**	Система тестування працює у повному обсязі, забезпечуючи перевірку коду учнів незалежно від тарифного плану.
Статистика та моніторинг	Обмежено учасниками Простору	Вчитель отримує зведену таблицю успішності лише для тих учнів (до 10 осіб), які додані до Простору. Результати інших учнів необхідно перевіряти вручну через їхні публічні профілі.

Таким чином, використання функціоналу «Простір» є доцільним для індивідуальної роботи або малих гуртків, тоді як для масового навчання без додаткового фінансування рекомендується використовувати методику роботи з відкритим архівом задач.

2.2. Добір та диференціація олімпіадних задач II етапу UOI 2022-2025 років

Для побудови ефективної та релевантної навчальної програми підготовчого курсу нами було здійснено детальний аналіз комплектів завдань II (районного) етапу Всеукраїнської учнівської олімпіади з інформатики за останні чотири роки (2022–2025). Метою аналізу було виявлення стійких тематичних блоків, оцінка рівня складності задач та визначення необхідної знаннєвої бази для їх розв’язання.

Джерельною базою аналізу слугували офіційні методичні рекомендації організаторів олімпіад, тексти завдань та авторські розбори (туторіали).

У поточному навчальному році завдання I етапу (старого II) [56] характеризувалися чітким поділом на рівні складності. Базовий рівень вимагав

від учасників навичок роботи з арифметичними операціями та простими умовами, тоді як задачі високого рівня передбачали знання алгоритмів оптимізації (таблиця 2.2). Як зазначається в офіційному розборі завдань 2025 року [57], задача H («Виставка») вимагала застосування методу префіксних сум для швидкого обчислення суми на підвідрізках, що підтверджує необхідність включення цієї теми до програми підготовки.

Таблиця 2.2

Аналіз задач UOI I етап (районний) сезону 2025/2026 років

Рівні	**Задачі**	**Теми програмування**	**Знаннєва база**
Базовий рівень	A, B, D	Введення-виведення даних Арифметика та логіка Умовні оператори Робота з рядками та типами даних	map(int, input().split())) if/else прості формули (периметр, площа) перевірка простих умов розрізняти рядкові та числові типи даних
Середній рівень	C, E, F	Масиви (списки), Цикли Жадібні алгоритми Матриці та координатна площина Теорія чисел (основи)	Лінійний пошук Перебір елементів масиву Поняття подільності та кратності Операція взяття остачі від ділення (%) Властивості діагоналей Розуміння індексації в 2D масивах
Високий рівень	G, H	Префіксні суми Метод «Зсувного вікна« Сортування та рядки Стек Оцінка складності алгоритмів	Алгоритми O(N), O(NlogN), O(N^2) Структура даних LIFO (Last In, First Out) Порядок елементів

Особливістю завдань сезону 2024/2025 років є відхід від складного кодування (використання дерева відрізків, графів тощо) на користь *математичного мислення, вміння помічати інваріанти та будувати конструктивні доведення*. Наприклад, задача D («М’ячі та контейнери») [59, 1]

вимагає розуміння інваріанту – сумарна кількість кульок не змінюється при операціях. Розв'язок зводиться до перевірки подільності на 3 та використання формули з модулями. В задачі E («Красивий рядок») [59, 1] доводиться, що завжди вигідно рухатися вперед, якщо світлофор зелений, а очікування не покращує результат. Рішення моделює рух Сакурако за O(n). Такі задачі перевіряють здатність учасників знаходити прості та елегантні рішення для, на перший погляд, складних процесів (таблиця 2.3).

Розробники завдань сезону 2023/2024 років акцентували увагу на конструктивні алгоритми та симуляційні задачі. Особливістю стало використання частотних масивів та евристичних підходів (таблиця 2.4). У задачі F («Подарунки з чемпіонату«) учасникам необхідно було реалізувати складну логіку пріоритетів при виборі елементів [58, 1], що вимагало навичок структурування коду та роботи з вкладеними умовами. Наявністю завдань на роботу з двовимірними масивами та складними симуляціями процесів у часі («Підберіть кольори», задача H) вимагала від учасників знання бінарного пошуку по відповіді або рекурсії з запам'ятовуванням [58, 2], що є ознакою переходу до алгоритмів вищої складності.

Таблиця 2.3

Аналіз задач UOI II етап (районний) сезону 2024/2025 навчального року

Рівні	**Задачі**	**Теми програмування**	**Знаннєва база**
Базовий рівень	A, B, C	Введення-виведення даних Базова математика та геометрія Умовні оператори (if/else) Робота з координатами	map(int, input().split()) Арифметичні операції (+, -, *, **) Прості формули (площа, відстань) Порівняння чисел (<=, >=)

Рівні	Задачі	Теми програмування	Знаннєва база
Середній рівень	D, E, F, G	Теорія чисел (подільність, остача) Циклічні процеси Робота з рядками та масивами Конструктивні алгоритми	Оператор остачі від ділення % Перевірка на парність (x % 2 == 0) Пошук патернів у рядках («11», «101») Сортування масивів .sort()
Високий рівень	H	Частотні масиви (Frequency array) Властивості паліндромів Оптимізація та мінімізація функцій Евристичний перебір	Підрахунок кількості входжень елементів Попередній розрахунок Робота з індексами (i та n−1−i) Алгоритмічна складність (уникнення O(N2)) «Жадібний» підхід з умовами

Аналіз завдань олімпіадного сезону 2022/2023 Завдання II етапу 2022 року вирізнялися акцентом на прикладну математику та симуляційні моделі. Особливу увагу було приділено уникненню помилок точності при роботі з дробовими числами та оптимізації маршрутів (таблиця 2.5).

Таблиця 2.4

Аналіз задач UOI II етап (районний) сезону 2023/2024 років

Рівні	Задачі	Теми програмування	Знаннєва база
Базовий рівень	A, B, C, D	Арифметика та прості формули Лінійні алгоритми Пошук мінімуму/максимуму Повний перебір (Brute force) для малих даних	int(input()), арифметичні оператори Переведення дат у дні (формула) Функції min(), max() Прості цикли для перебору варіантів

Рівні	Задачі	Теми програмування	Знаннєва база
			Логічне мислення (пошук математичної закономірності)
Середній рівень	E, F, G	Двовимірні масиви (матриці) Симуляція процесів (Step-by-step simulation) Сортування складних об'єктів Робота з рядками та словниками (Hash Maps)	Вкладені цикли (for i... for j...) Індексація в матрицях (grid[y][x]) Сортування з ключем (sort(key=lambda...)) Словники (dict) для зберігання властивостей об'єкта Реалізація складної умови з пріоритетами
Високий рівень	H	Бінарний пошук по відповіді Рекурсія Динамічне програмування (Мемоізація) Оптимізація алгоритмів	Алгоритм бінарного пошуку (low, high, mid) Розуміння, коли симуляція занадто повільна Рекурсивні функції з поверненням значень Кешування результатів (memo = {})

Характерним прикладом базового рівня стала задача C («Петрик та екзамен»), де перевірка умови успішності (51%) вимагала від учасників розуміння особливостей комп'ютерної арифметики. Замість ділення, яке призводить до втрати точності (float), рекомендувалося використовувати множення, зводячи задачу до цілочисельних операцій .

У завданнях середнього рівня, таких як задача E («Скрутні часи»), активно використовувалася модульна арифметика для симуляції періодичних процесів (робочі та вихідні дні) у циклі .

Високий рівень складності був представлений задачами на динамічне програмування та роботу з матрицями. Зокрема, задача G («Дорога в нове життя«) передбачала пошук оптимального шляху з мінімізацією витрат, що можна було реалізувати через метод динамічного програмування або жадібний алгоритм із сортуванням. Задача H («Шахова проблема«) вимагала навичок

роботи з двовимірними масивами та векторами руху (dx, dy) для перевірки стану шахової дошки (мат, пат) [60].

Таблиця 2.5

Аналіз задач UOI II етап (районний) 2022/2023 років

Рівні	Задачі	Теми програмування	Знаннєва база
Базовий рівень	A, B, C, D	Введення-виведення даних Арифметика та перетворення формул Умовні оператори (if-elif-else) Робота з відсотками (через цілі числа) Пошук мінімуму серед кількох значень	Функції input(), print(), int() Математичні оператори (+, -, *) Логічні оператори (and, or, ==) Функція min() Уникнення дробових чисел (float) шляхом множення
Середній рівень	E, F	Цикли (for, while) Модульна арифметика (остача від ділення) Симуляція процесів у часі Математичні прогресії	Оператор % (остача) для визначення циклічності Формула суми арифметичної прогресії (2n(n+1)) «Жадібний» підхід (Greedy) Обробка крайових випадків у циклах
Високий рівень	G, H	Динамічне програмування Теорія графів (пошук найкоротшого шляху) Двовимірні масиви (матриці) Вектори руху (Direction Arrays) Складна логіка та декомпозиція задачі	Сортування списків (.sort()) Поняття релаксації (оновлення кращого результату) Індексація масивів board[row][col] Використання dx, dy для опису ходів фігур Обробка виходу за межі масиву

На основі проведеного аналізу нами було сформовано перелік тем, які є критично важливими для успішної участі в олімпіаді (таблиця 2.6). Ці теми лягли в основу розробленої навчальної програми. Результати аналізу свідчать про те, що попри наявність складних задач (рівень G-H), основний масив балів

(рівень A-F) можна отримати, досконало володіючи базовими конструкціями мови програмування (умови, цикли, списки) та математичною логікою. Нами було обрано основні теми, які допоможуть учню у підготовці.

Таблиця 2.6

Теми, які треба врахувати під час підготовки

Тема	Підтеми	Частота появи	Складність реалізації мовою Python
Лінійні алгоритми	Зчитування в рядок (input) Зчитування масиву чисел (map) Форматований вивід (f-strings) Робота з типами (int, float)	**Дуже висока**	Дуже низька
Арифметика (Теорія чисел)	Виділення цифр числа (% 10, // 10) Парність (% 2) Дільники числа НСД і НСК (math.gcd) Перевірка на простоту	**Дуже висока**	Низька
Умовні алгоритми	Вкладені умови (if-elif-else) Логічні операції (and, or, not) Задачі на координати (чверті, шахівниця)	**Висока**	Низька
Цикли	Пошук суми/добутку Пошук мінімуму/максимуму (min/max) Лінійний пошук елемента Генерація чисел у діапазоні (range)	**Висока**	Середня

Тема	Підтеми	Частота появи	Складність реалізації мовою Python
Рядки та символи	Зрізи (s[::-1], s[1:5]) ASCII коди (ord, chr) Пошук підрядка (in, .find) Заміна та підрахунок (.replace, .count)	**Середня**	Низька
Масиви (списки)	Прості операції: додати, видалити Сортування (.sort(), sorted()) Реверс масиву Частотні масиви (підрахунок кількості входжень)	**Середня**	Низька
Геометрія (базова)	Периметр/Площа прямокутників Відстань між точками (Манхеттенська та Евклідова) Перетин відрізків (на прямій)	**Низька**	Середня
Графи та матриці (початковий рівень)	Сусідство клітинок у двовимірному масиві (сітка) Лабіринти (дуже прості, без BFS/DFS) • *Класичні графи (дерева, цикли) відсутні*	**Дуже низька**	Висока

2.3. Структура та зміст навчальної програми підготовчого курсу для учнів 10-11(12) класів

Навчальна програма факультативного курсу (гуртка) **«Основи олімпіадного програмування»** (додаток А)призначена для учнів **10-11(12) класів**, які не мають попереднього досвіду участі в змаганнях з інформатики, але прагнуть опанувати навички розв’язання алгоритмічних задач.

Програма розроблена з урахуванням сучасних тенденцій розвитку ІТ-освіти, вимог Всеукраїнських учнівських олімпіад з інформатики (UOI) та орієнтована на підготовку до ІІ (районного) етапу змагань.

Головною метою курсу є формування в учнів алгоритмічного мислення, розвиток навичок програмування мовою Python, ознайомлення з основними класами олімпіадних задач та підготовка до успішної участі в змаганнях районного рівня.

В основу програми покладено **практико-орієнтований підхід** з використанням автоматизованої системи перевірки розв'язків.

Завдання курсу:

Надати учням знання про:

- структуру та регламент проведення олімпіад з інформатики;
- синтаксис та особливості мови програмування Python в контексті спортивного програмування;
- базові алгоритмічні конструкції (лінійні, розгалужені, циклічні);
- типові методи розв'язання задач (жадібні алгоритми, префіксні суми, метод двох вказівників).

Сформувати вміння учнів:

- працювати з автоматизованою системою перевірки Eolymp (відправка рішень, аналіз результатів тестування);
- розробляти ефективні алгоритми для задач різного рівня складності (від A до H);
- тестувати та налагоджувати власний програмний код;
- працювати в умовах обмеженого часу.

Характеристика структури і змісту навчальної програми.

Програма розрахована на **27 годин** та має модульну структуру, побудовану за принципом «від простого до складного» (спіральна модель навчання).

Зміст курсу базується на детальному аналізі завдань ІІ етапу Всеукраїнської олімпіади з інформатики за 2022–2025 роки. Програма складається з **6 розділів**:

1. Вступ. Ознайомлення з середовищем тестування. Арифметика (3 год). Цей розділ спрямований на «швидкий старт» в олімпіадному програмуванні за мінімально необхідний час. Перша година присвячена технічним аспектам (реєстрація на Eolymp, структура програми, ввід/вивід), а дві наступні – базовій математиці (стандартні операції, операції ділення націло та остачі) та геометрії (обчислення периметрів, площ, відстаней у координатному просторі). Основна мета – навчити учнів взаємодіяти з автоматизованою системою перевірки.

2. Умовні оператори та конструктивні задачі (4 год). Даний розділ передбачає розвиток логічного мислення та опануванні розгалужених алгоритмічних структур. Учні вивчають синтаксис умовних операторів, булеву алгебру для побудови складених умов та оптимізації логічних перевірок. Окремий акцент робиться на розв’язанні конструктивних задач – завдань, що вимагають не простого кодування формули, а попереднього логічного аналізу, розгляду граничних випадків та пошуку інваріантів (наприклад, аналіз цифр числа, принцип Діріхле, задачі на пакування), які часто зустрічаються на олімпіадах

3. Циклічні алгоритми та складні симуляції (5 год). Розділ присвячено моделюванню процесів, що повторюються або розгортаються у часі. Учні опановують циклічні конструкції для обробки послідовностей та реалізації ітераційних обчислень. Важливою складовою є вивчення модульної арифметики для роботи з періодичними процесами (дні тижня, циклічні зміни станів). Розглядаються задачі на покрокову симуляцію («step-by-step»), де необхідно програмно відтворити алгоритм дій, описаний у легенді задачі, а також основи роботи з вкладеними циклами для генерації числових та символьних патернів.

4. Масиви, рядки та лінійні алгоритм (5 год). У цьому розділі здійснюється перехід до роботи зі структурованими типами даних. Вивчаються методи створення, наповнення та обробки одновимірних масивів (списків) і рядків. Ключовими алгоритмічними темами є лінійний пошук (знаходження

мінімуму, максимуму, елемента за умовою), техніка частотного аналізу (підрахунок кількості входжень елементів) та реалізація «жадібних» підходів на масивах. Учні вчаться ефективно використовувати дані, що зберігаються в пам’яті, та розв’язувати задачі на пошук підрядків і закономірностей у текстових даних.

5. Сортування, вказівники, префіксні суми (6 год). Розділ охоплює алгоритми підвищеної складності, необхідні для розв’язання задач високого рівня (типу G-H). Розглядаються методи сортування даних та їх застосування для спрощення пошуку. Вивчається метод «двох вказівників» для ефективної обробки масивів за один прохід. Особлива увага приділяється техніці префіксних сум, що дозволяє оптимізувати обчислення суми на відрізках, а також основам роботи з двовимірними масивами (матрицями) та векторами руху для розв’язання задач на координатній площині та ігрових полях (наприклад, шахових).

6. Підсумковий контроль (4 год). Завершальний розділ спрямований на комплексну перевірку набутих компетентностей та психологічну підготовку до змагань. Навчальна діяльність організована у формі симуляції реальної олімпіади (віртуальні завдання) з обмеженням часу та забороною використання допоміжних матеріалів. Важливим елементом змісту є методика «Дорішування» – детальний розбір авторських розв’язків та аналіз типових помилок, що дозволяє закріпити знання та сформувати стратегію дій під час реального турніру.

Особливості організації освітнього процесу.

Навчання здійснюється у змішаному форматі з використанням хмарного сервісу **Eolymp** [61]. Методика проведення занять базується на моделі **«Прискорене вивчення – Аналіз – Дорішування»**:

- **Прискорене вивчення:** розв’язування добірки задач в умовах обмеженого часу (60 хв), що імітує реальну олімпіаду.
- **Аналіз:** розбір типових помилок та авторських розв’язків учителем.

- **Дорішування:** можливість здати нерозв’язані задачі після завершення основного часу для закріплення матеріалу.

На основі тематичного плану вчитель розробляє календарно-поурочний план, в якому зазначає обсяг навчального матеріалу. При цьому слід враховувати поєднання теоретичної частини занять та практичних робіт. У процесі засвоєння змісту курсу передбачене використання різноманітної наочності та інформаційно-комунікаційних технологій. Навчальна програма орієнтована на розвивальний характер занять. Виконання програми забезпечується високим рівнем підготовки кожного заняття та використанням актуальної бази задач минулих років. Вчитель має право вносити зміни в тематичний план до 20% часу залежно від рівня підготовки групи.

Під час навчальних занять необхідно дотримуватися вимог охорони праці, організації робочого місця учнів, здійснювати контроль за вивченням та виконанням ними правил безпеки праці, протипожежної безпеки, виробничої санітарії та гігієни праці.

Очікувані результати: після завершення курсу учні повинні:

- розуміти принципи роботи тестуючих систем;
- вміти розв’язувати задачі базового (A-C) та середнього (D-F) рівнів складності;
- мати уявлення про підходи до розв’язання задач високого рівня (G-H);
- бути психологічно готовими до участі в II етапі Всеукраїнської учнівської олімпіади з інформатики.

2.4. Організація практичних занять: модель “Прискорене вивчення – Аналіз – Дорішування”

Апробація розробленої методики розпочалася з впровадження першого **змістового модуля «Вступ. Арифметика»** (згідно з календарно-тематичним плануванням). Організація навчального процесу базувалася на циклічній моделі

«Прискорене вивчення – Аналіз – Дорішування», що дозволило інтенсифікувати набуття практичних навичок учнями.

Упродовж першого тижня було проведено три практичні заняття, кожне з яких мало чітку структуру та спрямованість на формування конкретних компетентностей.

На **першому занятті** («Знайомство з Eolymp. Структура програми») ключовим завданням було подолання бар'єра входу до автоматизованих систем перевірки. Замість традиційного вивчення синтаксису на уроках інформатики, учням було запропоновано концепцію «*Чорної скриньки*», де програма розглядається як інструмент, що перетворює вхідні дані на вихідні без зайвого діалогу з користувачем.

Під час першого етапу (***Прискорене вивчення***) учні виконували задачі рівня А («Борги», «Біткопійки»), що вимагали мінімального коду, але суворого дотримання формату введення-виведення. Було акцентовано на відмові від текстових підказок у функції input(), що є типовою помилкою новачків.

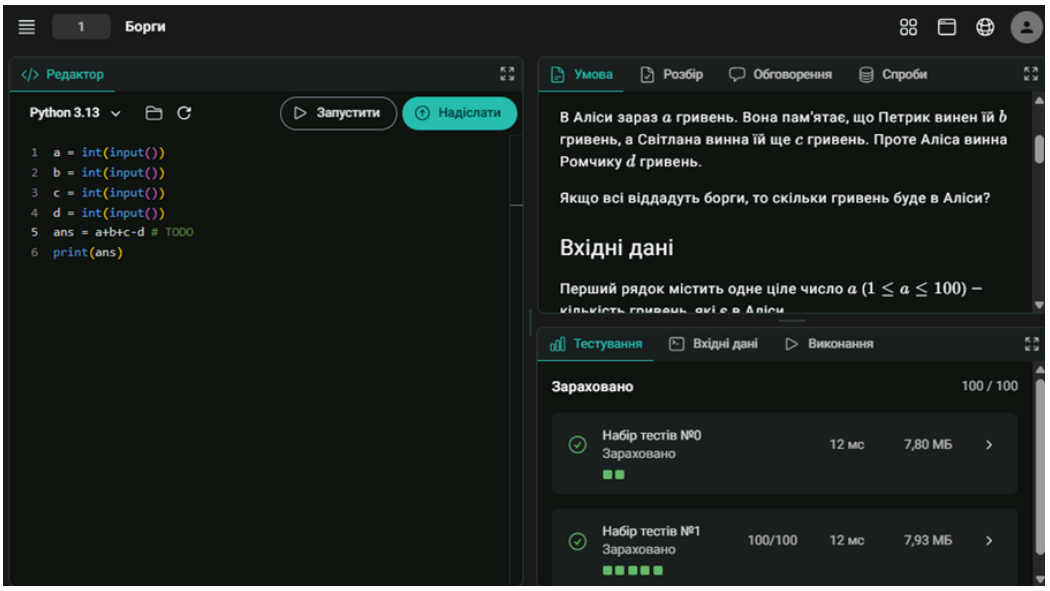

Рис. 2.5. Процес розв'язання та здачі завдання «Борги» учнем на платформі Eolymp

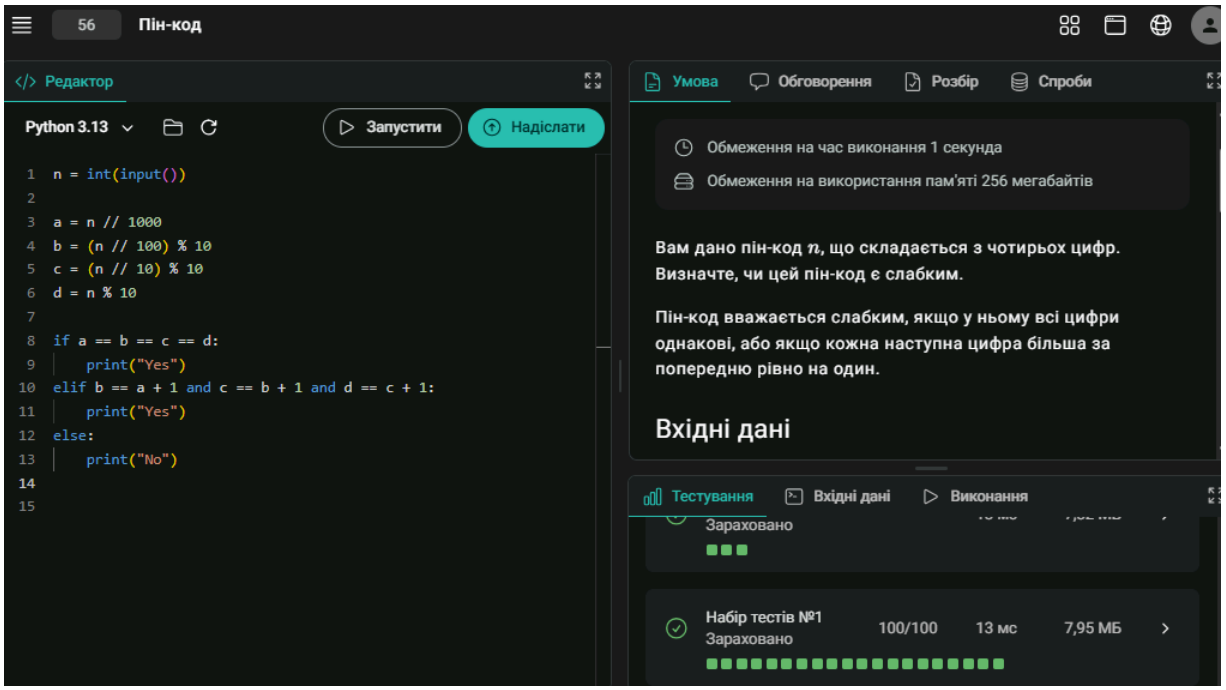

Рис. 2.6. Фрагмент коду задачі «Пін-код» з використанням операторів цілочисельного ділення

На **другому занятті** («Цілочисельна арифметика») фокус змістився на специфічні оператори Python // та %. Замість абстрактної теорії, учні одразу застосовували ці оператори для розв'язання прикладних задач: розбиття числа на розряди та знаходження координат на шахівниці (задача «Тура»). Використання методу демонстрації екрану вчителя дозволило візуалізувати різницю між звичайним діленням (/) та цілочисельним, що стало основою для розв’язання задачі «Середнє арифметичне» без втрати точності.

Третє заняття («Математичне моделювання та геометрія») було присвячене роботі з бібліотеками, зокрема модулем math. Учні опановували навички формалізації геометричних задач («Координати», «Прямокутник»), переходячи від математичних формул на папері до їх програмної реалізації.

Важливою складовою методики є навчання учнів інтерпретації автоматичних вердиктів системи Eolymp (***Аналіз***). У ході занять вчитель виконував роль ментора, аналізуючи не лише код, а й логіку мислення учнів.

Було виділено типові помилки:

Compilation Error (CE): синтаксичні помилки, пропущені дужки, неправильна змінна і т.д.

Wrong Answer (WA): наявність зайвого тексту у виведенні (наприклад, «Відповідь:») або використання некоректних типів даних (плутанина між int та float).

Завершальним етапом кожного тижневого циклу є ***Дорішування***. Учні, які не встигли розв'язати всі задачі під час уроку (наприклад, задачі підвищеної складності або додаткові, такі як «Цифри числа»), мали можливість завершити їх пізніше.

Це дозволило реалізувати індивідуалізований підхід: сильніші учні переходили до задач наступного рівня (наприклад, задачі на логіку з наступного модуля), тоді як учні, що потребували більше часу, закріплювали базові навички в комфортному темпі. Домашні завдання, сформульовані в КТП, виконувалися саме в цьому режимі, що забезпечило безперервність навчального процесу.

Впровадження моделі «Прискорене вивчення – Аналіз – Дорішування» вже на першому тижні навчання продемонструвало підвищення мотивації учнів та швидку адаптацію до вимог олімпіадного програмування.

2.5. Моніторинг результативності підготовки учнів

Невід’ємною складовою розробленої методики є систематичний моніторинг навчальних досягнень, який дозволяє об’єктивно оцінити рівень сформованості алгоритмічного мислення та готовності учнів до змагальної діяльності.

На відміну від традиційного контролю знань, в олімпіадному програмуванні моніторинг має виконувати дві функції: **діагностичну** (виявлення прогалин у знаннях) та **прогностичну** (оцінка шансів на успіх у реальному турнірі).

Згідно з календарно-тематичним планом, наприкінці першого тижня навчання було проведено підсумкове тренувальне змагання за темою першого розділу «**Вступ. Ознайомлення з середовищем тестування. Арифметика**». Метою заходу було створення умов, максимально наближених до реального ІІ етапу (нині І етапу) Всеукраїнської олімпіади:

- 60 хвилин на 4 задачі;
- заборона комунікації між учасниками;
- використання системи IOI (часткові бали).

Набір завдань для змагання був сформований із задач архіву UOI попередніх років (зокрема, розглянутих на уроці), а також з відкритого простору Basecamp, щоб перевірити не пам'ять, а розуміння алгоритмів.

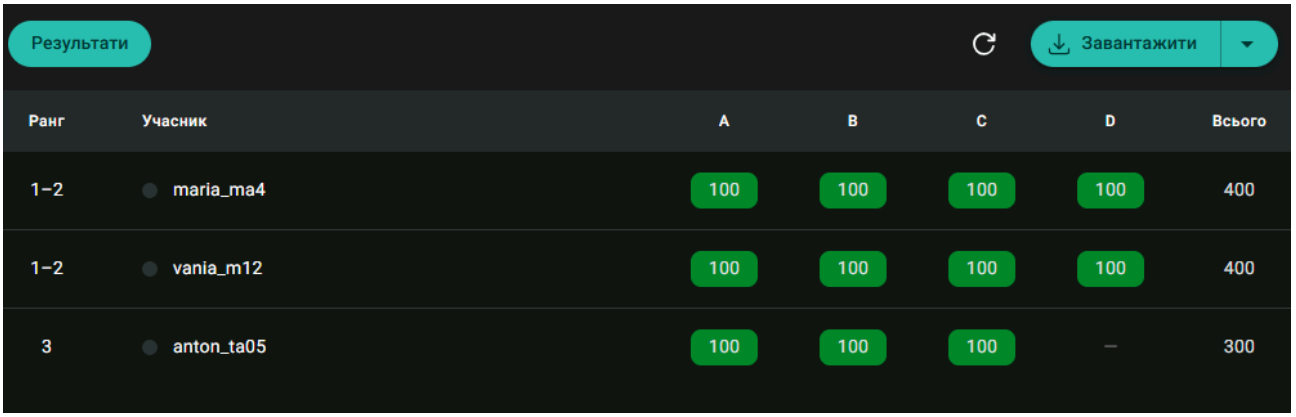

Рис. 2.7. Загальна таблиця результатів підсумкового змагання першого тижня

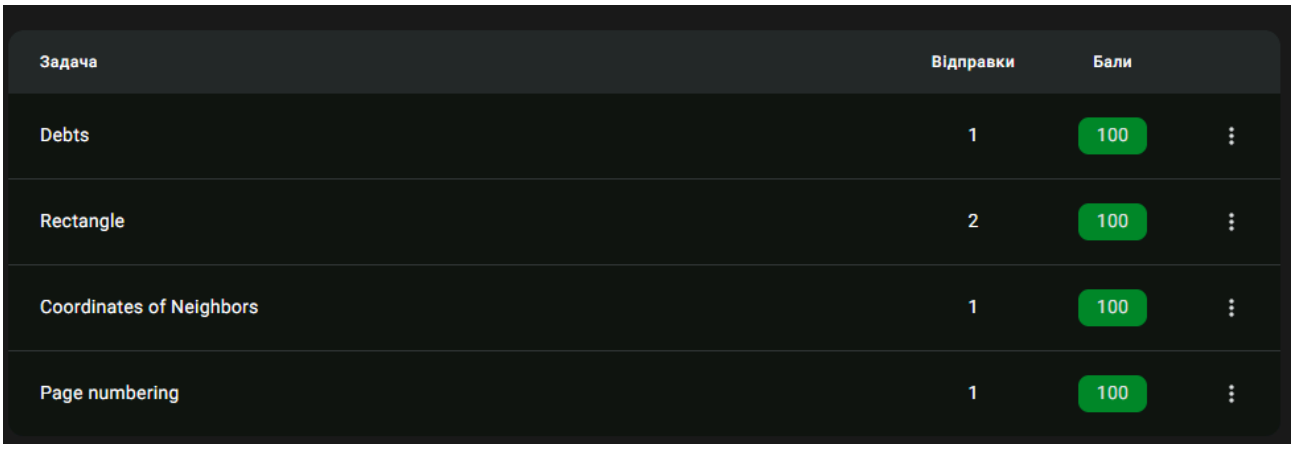

Рис. 2.8. Статистика відправки розв'язків конкретного учня

Використання автоматизованої системи Eolymp дозволило отримати певну статистику успішності групи. Аналіз результатів здійснювався за такими критеріями:

Коефіцієнт розв'язуваності – відношення кількості успішних спроб (AC) до загальної кількості учасників. Високий показник у задачах рівня A підтвердив ефективність вступних занять.

Кількість спроб – аналіз системи показав, що незначна частина учнів робить декілька спроб відправки рішень без попереднього тестування (*сліпий метод*). Це свідчить про достатнє розуміння учнями важливості етапу тестування програмного коду.

Типологія помилок – помилки, які були виявлені під час кодування:

- *Compilation Error (CE):* майже в усіх учасників (переважно синтаксичні помилки або некоректні відступи в Python) під час вирішення задач на уроці. Відсутні на етапі віртуального тренування.
- *Time Limit Exceeded (TLE):* відсутні на цьому етапі, оскільки задачі лінійні.
- *Wrong Answer (WA):* найбільш поширена помилка, пов'язана з неуважним читанням умови або хибною математичною моделлю задачі.

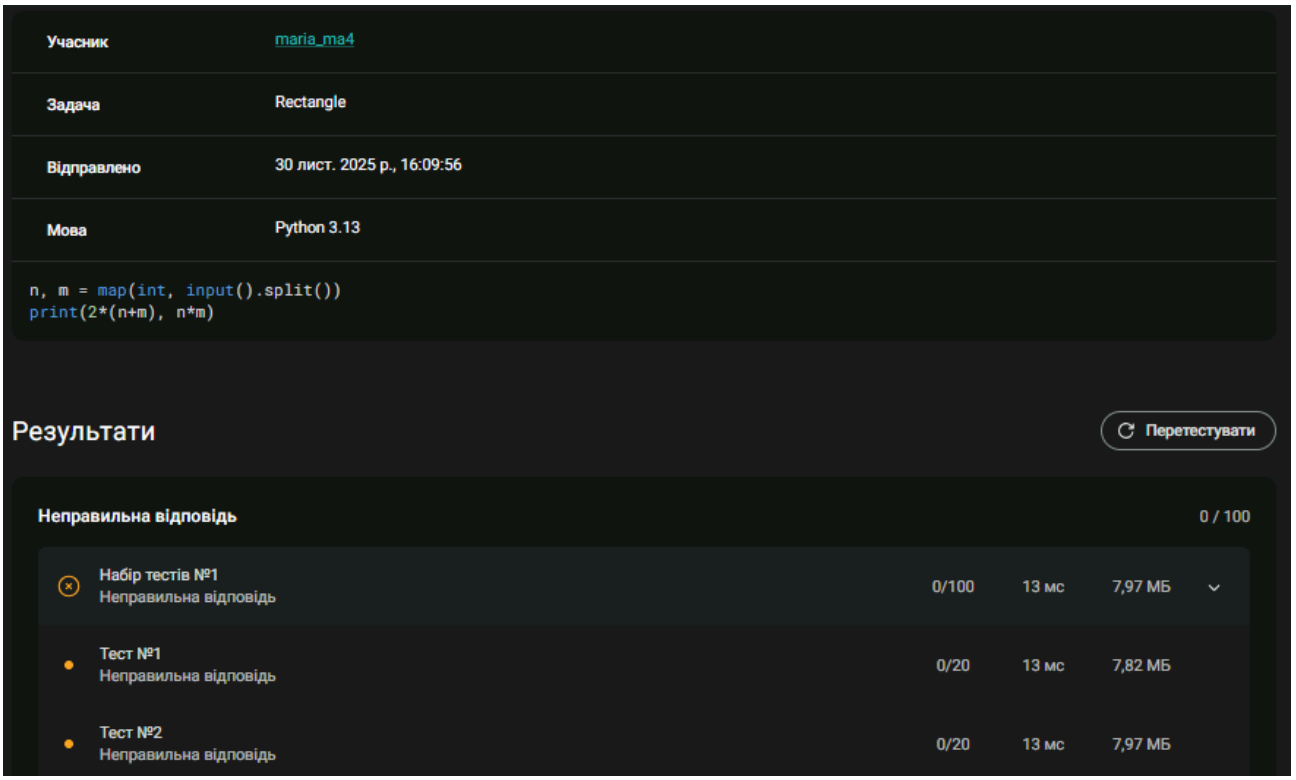

Рис. 2.9. Детальний результат відповіді на роботу учня

Результати моніторингу стали основою для диференціації домашнього завдання в режимі **Дорішування (Upsolving)**:

- Учні, які розв'язали 100% задач змагання, отримали доступ до додаткового блоку задач підвищеної складності (рівень B-C).
- Учні, які припустилися помилок, отримали завдання «Робота над помилками» з обов'язковим аналізом авторського коду (code review) спільно з вчителем.

Такий підхід перетворює жорсткий контроль успішності на засіб навігації у навчальному процесі, що відповідає принципам особистісно-орієнтованого навчання. Автоматизація перевірки через Eolymp дозволила виключити суб'єктивний фактор оцінювання та суттєво зекономити час вчителя для якісного аналізу результатів.

Висновки до розділу 2

У другому розділі здійснено розробку та обґрунтування методичних засад індивідуалізованої підготовки учнів 10–11(12) класів до олімпіад з програмування. За результатами проведеного дослідження зроблено наступні висновки.

Доведено, що використання хмарної платформи **Eolymp** є оптимальним технічним рішенням для організації підготовки початківців. Її функціонал (автоматизована перевірка, доступ до архіву UOI, режими змагання та дорішування) дозволяє реалізувати принцип миттєвого зворотного зв’язку, усунути суб’єктивність оцінювання та забезпечити диференціацію навчання в умовах великої наповнюваності груп.

На основі детального аналізу завдань II (районного) етапу Всеукраїнської олімпіади з інформатики за 2022–2025 роки виявлено ключові тематичні блоки та типологію задач. Це стало підґрунтям для розробки навчальної програми спецкурсу **«Основи олімпіадного програмування»** (27 годин). Програма побудована за модульним принципом, що дозволяє гнучко адаптувати її під

рівень підготовки учнів, зосереджуючись на критично важливих темах (лінійні алгоритми, умовні оператори, цикли, масиви).

Запропоновано та апробовано авторську модель організації практичних занять **«Прискорене вивчення – Аналіз – Дорішування»**. Встановлено, що поєднання роботи в умовах жорсткого обмеження часу (імітація олімпіади) з наступним детальним аналізом та безлімітним дорішуванням дозволяє ефективно поєднати спортивну складову (стресостійкість, тайм-менеджмент) з глибоким засвоєнням матеріалу в індивідуальному темпі.

Розроблено систему діагностики результативності, яка базується не на традиційному оцінюванні, а на аналізі вердиктів тестуючої системи (AC, WA, CE) та динаміки спроб. Це дозволяє вчителю реалізувати функцію ментора: оперативно виявляти прогалини в знаннях конкретних учнів та коригувати їхню індивідуальну освітню траєкторію через надання різнорівневих завдань (базових для закріплення або ускладнених для поглиблення). Такий підхід є оптимальним під час підготовки до II (нині I, районного) етапу Всеукраїнської олімпіади з інформатики.

Отже, розроблені методичні засади створюють цілісну систему підготовки, яка трансформує роль вчителя з транслятора знань на організатора персоналізованого освітнього середовища, що є критично важливим для успішної участі школярів в інтелектуальних змаганнях.

ВИСНОВКИ

У кваліфікаційній роботі здійснено теоретичне обґрунтування, розробку та апробацію методики індивідуалізованої підготовки учнів старших класів до олімпіад з програмування. Результати дослідження дають підстави сформулювати основні висновки.

На основі аналізу науково-педагогічної літератури *з'ясовано*, що олімпіадне програмування є потужним засобом розвитку алгоритмічного та критичного мислення старшокласників. Встановлено, що традиційні класно-урочні методи є недостатньо ефективними для підготовки до інтелектуальних змагань через високу гетерогенність рівня підготовки учнів. Доведено необхідність переходу до моделі індивідуалізованого навчання, де ключову роль відіграє тьюторський супровід, спрямований на побудову персональної освітньої траєкторії та надання якісного зворотного зв'язку.

Визначено, що ефективність навчання алгоритмізації учнів 10–11(12) класів залежить від урахування їхніх вікових особливостей: професійного самовизначення, потреби у самореалізації та здатності до рефлексії. Обґрунтовано, що дидактичні принципи індивідуалізації та персоналізації в умовах профільної школи найефективніше реалізуються через поєднання самостійної роботи в цифровому середовищі з менторською підтримкою вчителя.

Обґрунтовано вибір хмарної платформи **Eolymp** як оптимального технічного засобу для організації індивідуальної роботи. Доведено, що функціонал платформи (автоматизована система перевірки, доступ до архіву завдань, режими змагання та дорішування) дозволяє реалізувати об'єктивний контроль знань, забезпечити миттєвий зворотний зв'язок (вердикти системи) та звільнити час вчителя для індивідуальних консультацій.

Здійснено детальний аналіз завдань ІІ (районного) етапу Всеукраїнської учнівської олімпіади з інформатики за 2022–2025 роки. Виявлено стійкі тематичні блоки (лінійні алгоритми, умовні оператори, цикли, масиви) та типові

алгоритмічні конструкції, що стало основою для відбору змісту навчання. Встановлено, що для успішного виступу на початковому етапі критично важливим є не лише знання синтаксису, а й навички математичного моделювання та оптимізації коду.

Розроблено навчальну програму спецкурсу **«Основи олімпіадного програмування»** обсягом 27 годин, структуровану за модульним принципом (від базової арифметики до префіксних сум та сортування). Впроваджено авторську методику організації занять за моделлю **«Прискорене вивчення – Аналіз – Дорішування»**, яка поєднує роботу в умовах жорсткого обмеження часу (імітація змагання) з можливістю глибокого опрацювання матеріалу в індивідуальному темпі.

Складено методичні рекомендації для вчителів інформатики, що включають алгоритми реєстрації та налаштування контестів на платформі Eolymp, сценарії проведення занять та критерії діагностики готовності учнів. Запропонована методика дозволяє трансформувати роль вчителя з транслятора знань на тьютора, що підвищує мотивацію учнів та результативність їхньої участі в олімпіадах.

Таким чином, мету роботи досягнуто, поставлені завдання виконано. Результати дослідження можуть бути впроваджені в практику роботи закладів загальної середньої освіти для організації гурткової та факультативної роботи з інформатики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кривонос О. М. Підготовка студентів до участі у змаганнях з олімпіадного програмування. *Вісник Житомирського державного університету*. 2024. С. 115–117.

2. Носаченко Д. Інтеграція інноваційних підходів у підготовку до олімпіад з програмування. *Освітній дискурс*. 2023. С. 101.

3. Волошина Т. та ін. Платформи та системи автоматизованої перевірки завдань з програмування: аналіз, критерії добору та приклад використання. *Відкрите освітнє е-середовище сучасного університету*. 2020. № 8. С. 154–164.

4. Вапнічний С. Д., Жуковський С. С. Проведення шкільних олімпіад з інформатики в умовах пандемії. *Вісник Житомирського державного університету імені Івана Франка. Педагогічні науки*. 2022. Вип. 110. С. 67–84.

5. Опанасюк Н. Розвиток алгоритмічного мислення учнів при вивченні програмування. *Наукові записки*. 2024.

6. Sweller J. Cognitive load during problem solving: Effects on learning. *Cognitive science*. 1988. Vol. 12, no. 2. P. 257–285.

7. Шмирук Н. І. Впровадження інноваційних підходів та використання інтерактивних технологій у роботі з обдарованими учнями. *Обдаровані діти – скарб нації!*: матеріали III Міжнар. наук.-практ. конф. Київ: Інститут обдарованої дитини НАПН України, 2022. С. 1031.

8. Дем'янчук Ю. В. Підтримка обдарованих учнів через міжпредметну взаємодію: технології та інформатика в сучасному освітньому середовищі ЗЗСО. *Обдарованість: методи діагностики та шляхи розвитку*: матеріали конф. 2025. С. 247.

9. Кривонос О. М., Андрощук М. В. Інтерактивні платформи для вивчення програмування: як школярі 6-11 класів сприймають і використовують онлайн-ресурси. *Педагогічна наука і освіта XXI століття*. 2025. № 4. С. 156–170.

10. Нєшкова А. М. Інформатика в НУШ: виклики та перспективи. *Сучасна школа*. 2023.

11. Khmelevsky Y., Chidlow K. Students Programming Competitions as an Educational Tool and a Motivational Incentive to Students. *arXiv preprint arXiv:2105.15136*. 2021.

12. Yuen K. K., Liu D. Y., Leong H. V. Competitive programming in computational thinking and problem solving education. *Computer Applications in Engineering Education*. 2023. Vol. 31, no. 4. P. 850–866.

13. Сікан А., Кривонос О. М. Підготовка учнів до участі в олімпіадах з інформатики та інформаційних технологій з використанням Інтернет-ресурсів. *Інформаційні технології в освіті*. 2024. С. 158–161.

14. Makhrovska N. A., Pohromska H. S. Застосування онлайн змагань з програмування в системі практичної підготовки студентів спеціальності «комп'ютерні науки«. *Information Technologies and Learning Tools*. 2020. Vol. 79, no. 5. P. 260.

15. Бобир Р. В. Організація позакласної роботи з інформатики в контексті викликів сьогодення. *Педагогічні студії*. 2023.

16. Смірнова О., Баранова О. Особливості проведення ІІ та ІІІ етапів Всеукраїнських учнівських олімпіад з інформатики та інформаційних технологій. *Педагогічні обрії*. 2023. № 1. С. 29.

17. Носаченко Д. Вплив олімпіад з програмування на мотивацію учнів до вивчення інформатики. *Освіта. Інноватика. Практика*. 2022. Т. 10, № 8. С. 61–66.

18. Шулик Т., Соколова О. Становлення та розвиток інноваційної моделі STEM-освіти в Україні та США. *Гуманізація навчально-виховного процесу*. 2023. № 2 (104). С. 154–164.

19. Петрунько В., Соя О. Про участь українських учнів у міжнародних олімпіадах з інформатики. *Комп'ютер у школі та сім'ї*. 2024.

20. Плохотнюк В. Ю. До питання підтримки олімпіадного руху з інформатики в Україні. *Академічні студії*. 2023.

21. Стукалов О. С., Бойко О. П. Інформаційна підтримка навчання формальної логіки у старшій школі. *Інформаційні технології в освіті*. 2023.

22. Діденко П. Ю. Розвиток логічного мислення учнів старших класів при навчанні програмуванню. *Молодий вчений*. 2024.

23. Войтович І. С. та ін. Дистанційна підготовка учнів до участі в олімпіаді з інформаційних технологій. *Information Technologies in Education (ITE)*. 2025. № 1.57. С. 31–47.

24. Вербівський Д. С. Лабораторний практикум із застосування інноваційних технологій у навчанні інформатики. *Комп'ютерно-орієнтовані системи навчання*. 2025.

25. Про проведення Всеукраїнських учнівських олімпіад з навчальних предметів у 2025/2026 навчальному році: Наказ МОН України № 1165 від 20.08.2025 р.

26. Luo Z. Curriculum design of competitive programming: a contest-based approach. *arXiv preprint arXiv:2504.00533*. 2025.

27. Silvestre A. S. et al. A Multi-Label Classification Approach for Categorizing Beginner Programming Problems from Online Judges. *2024 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2024.

28. Nishanov A. et al. Methodology of Teaching Programming Science Through Online Platforms. *2024 IEEE 3rd International Conference on Problems of Informatics (PIERE)*. IEEE, 2024.

29. Sagyntay Y., Stoffová V., Katyetova A. Olympiad in Informatics – The Journey from the Domestic Round to International Competitions. *ICERI2023 Proceedings*. 2023.

30. Топузов О. М. та ін. Індивідуалізація навчання в умовах змішаної форми організації освітнього процесу в профільній старшій школі: метод. посібник. Київ, 2024.

31. Свінцицька Н. М. Потенційні можливості інформатичної освітньої галузі в напрямі діагностики та розвитку обдарованості учнів НУШ.

Обдарованість: методи діагностики та шляхи розвитку: матеріали конф. 2025. С. 776.

32. Грушко Р. С. Від теорії до практики: організаційно-педагогічні умови для формування цифрової компетентності на уроках інформатики. *Педагогічна інноватика: сучасність та перспективи*. 2024. № 4. С. 64–71.

33. Шаров С. Використання інформаційної системи для формування індивідуальної освітньої траєкторії. *ICISSE 2023 Proceedings*. 2023.

34. Лебідь О. М. Проєктування індивідуальної освітньої траєкторії майбутнього фахівця. Біла Церква, 2023. 133 с.

35. Мізюк В. А., Хижняк А. В., Хренова В. В. Використання адаптивних навчальних платформ для персоналізації дистанційного навчання. *Педагогічна Академія: наукові записки*. 2025. Вип. 14.

36. Lysenko S. Методика формування індивідуальної освітньої траєкторії фахівців комп'ютерного профілю в умовах неформальної освіти. *Problems of Engineer-Pedagogical Education*. 2023. No. 80.

37. Таган В. Соціально-психологічні фактори вибору професії у старшому шкільному віці. *Психологія і особистість*. 2025.

38. Шевченко В. Особливості розвитку пізнавальної самостійності старшокласників в процесі дослідницької діяльності. *Молодий вчений*. 2024. № 2 (126). С. 30–33.

39. Алєксєєва С. В. Актуальні проблеми дидактики в умовах інформатизації освіти: індивідуалізація навчання. *Наука і техніка сьогодні*. 2022. № 1.1. С. 18–27.

40. Сікора Я. Б. Персоналізація як підхід до навчання майбутніх ІТ-фахівців. *Цифрові технології в освіті*. 2023. С. 338–340.

41. Алексеєнко В. В., Гуменюк С. П. Використання цифрових інструментів для персоналізації навчання. *Тези доповідей*. 2023. С. 90.

42. Van Schoors R. et al. Design and development of a digital personalized learning track: Bridging the gap between textual and visual programming. *International Journal of Designs for Learning*. 2024. Vol. 15, no. 1. P. 74–95.

43. Сергеєва Н. Створення інтерактивних завдань на онлайн-сервісах – можливість забезпечення індивідуальної траєкторії розвитку дитини. *Особлива дитина: навчання і виховання*. 2025. № 117 (1). С. 216–227.

44. Prybora N. et al. Формувальне оцінювання у вищій освіті. *Continuing Professional Education: Theory and Practice*. 2025. Vol. 84, no. 3. P. 85–97.

45. Демидович О., Карапетян А. Agile-педагогіка: управління новими світоглядними викликами в освіті через динамічну адаптивність. *Актуальні питання гуманітарних наук*. 2024.

46. Горбань Л. В. Методики та технології педагогічного супроводу наукової освіти обдарованих учнів. *Освіта та розвиток обдарованої особистості*. 2024. № 2. С. 55–61.

47. Толок Д. Адаптивне тестування як інструмент організації дистанційного навчання. *Цифрові технології в освіті*: зб. наук. пр. Харків, 2024. Вип. 24. С. 142–150.

48. Фролова М., Дроздова Є., Козел В. Розвиток метакогнітивних навичок як чинник ефективного навчання програмуванню. *Psychology Travelogs*. 2025. № 2. С. 80–91.

49. Горбань Г. В. та ін. Автоматизація перевірки програмного коду з використанням сучасних технологій штучного інтелекту. *Вчені записки*. 2025.

50. Palahan S. PythonPal: Enhancing Online Programming Education through Chatbot-Driven Personalized Feedback. *IEEE Transactions on Learning Technologies*. 2025.

51. Дем'яненко, Н. «Архітектоніка тьюрингу у вищій школі.» Вісник Київського національного університету імені Тараса Шевченка. Педагогіка 2 (2018): 13-17.

52. Іреківна, Барієва Ельвіна, and Тимофій Васильович Топорков. «Методи тьюторської діяльності у навчанні інформатики.«

53. Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. Journal of Child Psychology and Psychiatry, 17(2), 89–100. DOI:https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

54. Zhou, Pinghong, et al. «The Effects of Scaffolding on Learning Outcomes in K-12 Programming Education: A Meta-Analysis.» 2023 Twelfth International Conference of Educational Innovation through Technology (EITT). IEEE, 2023.

55. Для організаторів змагань. URL: https://support.eolymp.com/uk/coaching (дата звернення 23.11.2025)

56. Нове Положення про олімпіади. URL: https://www.uoi.ua/news/new-regulations-2025 (дата звернення 23.11.2025)

57. Всеукраїнська олімпіада з інформатики 2025-2026 I етап. 8-11 класи 2 листопада 2025. URL: https://static.uoi.ua/tutorials/uoi2026-1s.pdf (дата звернення 23.11.2025)

58. Всеукраїнська олімпіада з інформатики 2023-2024, II етап 18 областей та місто Київ, 3 грудня 2023. URL: https://archive.uoi.ua/static/uoi-2-24-tutorials.pdf (дата звернення 23.11.2025)

59. Всеукраїнська олімпіада з інформатики 2025, II етап Україна, 1-ше грудня 2024. URL: https://static.uoi.ua/tutorials/ujgoi2025-vidbir-1.pdf (дата звернення 23.11.2025)

60. II етап Всеукраїнської олімпіади з інформатики 2022-2023 Україна, 4-те грудня 2022. URL: https://archive.uoi.ua/static/uoi-2-23-tutorials.pdf (дата звернення 23.11.2025)

61. Желудько Я.В. Тренувальний простір на Eolymp. *Підготовка до I етапу UOI*. URL: https://t05im6bomg.eolymp.space/ (дата звернення 30.11.2025)

ДОДАТКИ

Додаток А

Навчальна програма з підготовки до І етапу UOI для учнів-початківців 10-11(12) класів

ТЕМАТИЧНИЙ ПЛАН (27 годин)

№ з/п	Назва розділу	Кількість годин	
		Всього	З них на ПР*
1	**Розділ 1.** Вступ. Ознайомлення з середовищем тестування. Арифметика.	3	2
2	**Розділ 2.** Умовні оператори та конструктивні задачі	4	3
3	**Розділ 3.** Циклічні алгоритми та складні симуляції	5	4
4	**Розділ 4.** Масиви, рядки та лінійні алгоритми	5	4
5	**Розділ 5. С**ортування, вказівники, префіксні суми	6	5
6	**Розділ 6.** Підсумковий контроль (Віртуальна олімпіада)	4	4
	Разом	**27**	**22**

ПРОГРАМА

№	Год.	Очікувані результати (компетенції)	Зміст навчального матеріалу
1	3	**Розділ 1. Вступ. Ознайомлення з середовищем тестування. Арифметика**	
		Знаннєвий компонент: Знає структуру олімпіадної задачі, типи даних Python, методи швидкого введення (sys.stdin). **Діяльнісний компонент:** Вміє реалізовувати математичні моделі, працювати з геометрією та формулами. **Ціннісний компонент:** Розуміє важливість ефективності коду.	**Тема 1.1. Старт в Eolymp.** Реєстрація. Читання умов. Шаблон коду (import sys). Арифметичні оператори (//, %, **). **Тема 1.2. Математичне моделювання.** Задачі на формули. Геометрія на площині та в просторі. **ПР (Задачі рівня A-B):** *1. «Борги» (A) — лінійна формула.* *2. «Координати» (B) — відстань у 3D.* *3. «Прямокутник» (B) — площа рамки.* *4. «Середнє арифметичне» (B) — відновлення числа.*
2	4	**Розділ 2. Умовні оператори та конструктивні задачі**	
		Знаннєвий компонент: Розуміє логіку if-elif-else, булеву алгебру (and, or, not). **Діяльнісний компонент:** Вміє розв'язувати задачі на перебір варіантів, працювати з часовими інтервалами. **Ціннісний компонент:** Вміє аналізувати крайові випадки (Edge cases).	**Тема 2.1. Логічні розгалуження.** Порівняння чисел. Пошук min/max з 3+ значень. Вкладені умови. **Тема 2.2. Конструктивні задачі.** Задачі на принцип Діріхле, подільність, аналіз цифр. **ПР (Задачі рівня C-D):** *1. «Приготуйте домашнє завдання» (C) — часові інтервали.* *2. «Петрик та екзамен» (C) — відсотки без дробів.* *3. «Транспортна дилема» (D) — вибір оптимального шляху.* *4. «Конструктор» (D) — логіка з цифрами.* *5. «М'ячі та контейнери» (D) — інваріанти.*
3	5	**Розділ 3. Циклічні алгоритми та складні симуляції**	

№	Год.	Очікувані результати (компетенції)	Зміст навчального матеріалу
		Знаннєвий компонент: Розуміє використання циклів for/while, вкладені цикли. **Діяльнісний компонент:** Вміє моделювати покрокові процеси (Step-by-step simulation), працювати з періодичністю. **Ціннісний компонент:** Вміє розбивати складну задачу на підзадачі.	**Тема 3.1. Симуляція процесів.** Моделювання часових процесів. Робота з break, continue. **Тема 3.2. Модульна арифметика та вкладені цикли.** Періодичні події (світлофори, графіки). Генерація візерунків/матриць. **ПР (Задачі рівня E):** *1. «Сакурако запізнилася» (E) — парність/непарність часу.* *2. «Скрутні часи» (E) — складний графік роботи.* *3. «Дивний сон» (E) — рух по координатах.* *4. «День народження Сонечки» (E) — рух променя в матриці (вкладені цикли).*
4	5	**Розділ 4. Масиви, рядки та лінійні алгоритми**	
		Знаннєвий компонент: Знає методи списків (append, зрізи), роботу з рядками. **Діяльнісний компонент:** Вміє виконувати лінійний пошук, знаходити патерни в рядках, працювати з частотними характеристиками. **Ціннісний компонент:** Розуміє різницю між O(N) та O(N^2).	**Тема 4.1. Обробка одновимірних масивів.** Пошук підпослідовностей. Жадібні підходи на масивах. **Тема 4.2. Рядки та складна логіка.** Пошук підрядків. Реалізація складної бізнес-логіки (пріоритети). **ПР (Задачі рівня F):** *1. «Мобільна гра» (F) — пошук найдовшого ланцюжка.* *2. «Красивий рядок» (F) — аналіз бінарних рядків.* *3. «Подарунки з чемпіонату» (F) — симуляція складу футболок (пріоритети).* *4. «До чого ЗНО доводить» (F) — арифметична прогресія в циклі.*
5	6	**Розділ 5. Сортування, вказівники, префіксні суми**	

№	Год.	Очікувані результати (компетенції)	Зміст навчального матеріалу
		Знаннєвий компонент: Розуміє алгоритми сортування, метод двох вказівників, ідею префіксних сум. **Діяльнісний компонент:** Вміє оптимізувати перебір, працювати з двовимірними масивами, реалізовувати евристики. **Ціннісний компонент:** Готовність розв'язувати задачі підвищеної складності.	**Тема 5.1. Сортування та два вказівники.** Метод .sort(). Стратегія Min-Max. **Тема 5.2. Префіксні суми та динаміка.** Швидке обчислення суми на відрізку. Оптимізація маршруту. **Тема 5.3. Двовимірні масиви та рекурсія.** Координатна площина, шахи, бінарний пошук по відповіді. ***ПР (Задачі рівня G-H):*** *1. «Гарний масив» (G) — сортування + два вказівники.* *2. «Хто пройшов далі?» (G) — сортування об'єктів.* *3. «Виставка» (H) — префіксні суми + зсувне вікно.* *4. «Дорога в нове життя» (G) — жадібний алгоритм/динаміка.* *5. «Шахова проблема» (H) — 2D масиви, вектори ходів.* *6. «Підберіть кольори» (H) — бінарний пошук або рекурсія.*
6	4	**Розділ 6. Підсумковий контроль (Віртуальна олімпіада)**	
		Знаннєвий компонент: Комплексне застосування знань. **Діяльнісний компонент:** Робота в умовах стресу (обмежений час). Написання чистого коду. **Ціннісний компонент:** Навичка аналізу помилок.	**Тема 6.1. Пробна олімпіада (Virtual Contest).** Повноцінний контест на 2.5 - 3 години. Набір задач A-H. **Тема 6.2. Детальний розбір (Upsolving).** Аналіз авторських розв'язків. Оптимізація коду учнів. **ПР:** Контест змішаного типу.

Додаток Б

КАЛЕНДАРНО-ТЕМАТИЧНЕ ПЛАНУВАННЯ

№	Дата	Тема уроку	Зміст роботи	Домашнє завдання (ДЗ)
I	**ВСТУП. АРИФМЕТИКА (3 год)**			
1		Знайомство з Eolymp. Структура програми. Ввід/вивід даних	Реєстрація. Функції input(), print(), int(). **Практика:** *1. «Борги» (A).* *2. «Біткопійки» (A).*	Зареєструватися, дорішати задачі.
2		Цілочисельна арифметика	Операції //, %. **Практика:** *1. «Середнє арифметичне» (B).* *2. «Тура» (A).*	*Задача «Борги»*
3		Математичне моделювання та геометрія	Формули площі, відстані. Модуль math (корінь, степінь). **Практика:** *1. «Прямокутник» (B).* *2. «Координати» (B).*	*Задача «Міс М та квіти» (B).* Тренувальний тур на сайті
II	**УМОВНІ ОПЕРАТОРИ ТА ЛОГІКА (4 год)**			
4		Логічні розгалуження (if-else)	Синтаксис умов. Порівняння чисел. **Практика:** *1. «Приготуйте домашнє завдання» (C).* *2. «Петрик та екзамен» (C).*	*Задача «Перевірте календар!» (C).*

№	Дата	Тема уроку	Зміст роботи	Домашнє завдання (ДЗ)
5		Складна логіка (and, or). Пошук оптимального варіанту	Пріоритет операцій. Вибір min/max з 3-х значень. **Практика:** *1. «Транспортна дилема» (D).*	Скласти програму для сортування трьох чисел.
6		Конструктивні задачі (логіка без складного коду)	Аналіз вхідних даних, пошук інваріантів. **Практика:** *1. «Конструктор» (D).* *2. «М'ячі та контейнери» (D).*	Повторити ознаки подільності чисел.
7		Практикум з розв'язування логічних задач	**Практика (Закріплення):** Розв'язування задач, що викликали труднощі на попередніх уроках. Робота над помилками (Wrong Answer).	Дорішування контесту «Логіка».
III	**ЦИКЛІЧНІ АЛГОРИТМИ (5 год)**			
8		Цикл for. Накопичення сум	Функція range(), суматори. **Практика:** *1. «Скрутні часи» (E).*	Реалізувати обчислення суми чисел від 1 до N.
9		Модульна арифметика в циклах	Періодичність (парність кроків, дні тижня). **Практика:** *1. «Сакурако запізнилася» (E).* *2. «Дивний сон» (E).*	Задача «Біткопійки» (повторення).

№	Дата	Тема уроку	Зміст роботи	Домашнє завдання (ДЗ)
10		Цикл while. Моделювання процесів	Цикл з умовою. break, continue. **Практика:** *1. «До чого ЗНО доводить» (F)*	Закінчити задачу «ЗНО».
11		Вкладені цикли. Робота з координатною сіткою	Цикл у циклі. Генерація таблиць. **Практика:** *1. «Матричка» (C) (поняття діагоналей).*	Вивести на екран таблицю множення.
12		Складна симуляція (Рух у матриці)	**Практика:** *1. «День народження Сонечки» (E) (відбивання променя).*	Дорішування складних задач розділу.
IV	**МАСИВИ, РЯДКИ ТА ЛІНІЙНІ АЛГОРИТМИ (5 год)**			
13		Одновимірні масиви. Лінійний пошук	Списки в Python. Читання list(map(...)). Пошук min/max. **Практика:** *1. «Переїзд» (D).* *2. «Мобільна гра» (F) (пошук ланцюжка).*	Знайти суму елементів масиву.
14		Рядки та пошук підрядків	**Теорія:** Рядки як масиви символів. Методи рядків. **Практика:** *1. «Красивий рядок» (F).*	Знайти кількість голосних літер у слові.

№	Дата	Тема уроку	Зміст роботи	Домашнє завдання (ДЗ)
15		Частотні масиви та складна логіка	Підрахунок кількості входжень елементів. **Практика:** *1. «Подарунки з чемпіонату» (F) (частина 1 - інвентаризація).*	Продумати логіку заміни футболок.
16		Жадібні алгоритми на масивах	**Практика:** *1. «Подарунки з чемпіонату» (F) (частина 2 - реалізація замін).*	Дорішати задачу «Подарунки».
17		Практикум з лінійних алгоритмів	**Практика:** Закріплення навичок швидкого читання даних (sys.stdin) для великих масивів.	Підготовка до контрольної роботи.
V		**ВИСОКИЙ РІВЕНЬ (6 год)**		
18		Сортування та робота з об'єктами	Метод .sort(), lambda. **Практика:** *1. «Хто пройшов далі?» (G).*	Відсортувати список слів за довжиною.
19		Метод двох вказівників	Ефективний прохід по масиву з двох сторін. **Практика:** *1. «Гарний масив» (G).*	Дорішати задачу «Гарний масив».

№	Дата	Тема уроку	Зміст роботи	Домашнє завдання (ДЗ)
20		Префіксні суми (Теорія)	Швидке обчислення суми на відрізку. **Практика:** Розбір ідеї задачі «Виставка» (H). Реалізація простої версії.	Реалізувати масив префіксних сум.
21		Префіксні суми (Практика)	**Практика:** 1. Повна реалізація задачі «Виставка» (H).	Дорішування задачі.
22		Двовимірні масиви та вектори руху	dx, dy масиви для ходів. **Практика:** *1. «Шахова проблема» (H) (аналіз ходів фігур).*	Намалювати схему ударів коня.
23		Оптимізація та евристики	**Практика:** *1. «Дорога в нове життя» (G) або «Підберіть кольори» (H).*	Дорішування найскладніших задач.
VI		**ПІДСУМКОВИЙ КОНТРОЛЬ (4 год)**		
24		Віртуальна олімпіада (Частина 1)	**Змагання:** 3 задачі (Рівні A, B, C). Час: 60 хв. *Задачі обираються з архіву Eolymp, яких не було на уроках.*	Відпочинок.
25		Віртуальна олімпіада (Частина 2)	**Змагання:** 3 задачі (Рівні D, E, F). Час: 60 хв.	Відпочинок.

№	Дата	Тема уроку	Зміст роботи	Домашнє завдання (ДЗ)
26		Віртуальна олімпіада (Частина 3)	**Змагання:** 2 задачі (Рівні G, H). Час: 60 хв. *Для найсильніших учнів.*	Відпочинок.
27		Аналіз результатів (Upsolving)	Розбір авторських розв'язків. Нагородження переможців рейтингу групи.	Підготовка до реального І етапу.

Додаток В

План-конспект уроку № 1

«___» _____ 20___ року

Урок № 1

Тема: Знайомство з Eolymp. Структура програми. Ввід/вивід даних.

Мета:

- ✔ **навчальна:** ознайомити учнів з особливостями олімпіадного програмування та інтерфейсом платформи Eolymp; сформувати навички роботи з тестуючою системою (реєстрація, відправка розв'язку, аналіз вердиктів); навчити писати найпростіші лінійні програми мовою Python (введення/виведення даних, арифметичні операції).
- ✔ **розвиваюча:** розвивати алгоритмічне мислення, уважність до деталей (синтаксис, формат виводу), вміння працювати в умовах обмеженого часу.
- ✔ **виховна:** виховувати інформаційну культуру, дисциплінованість, наполегливість у пошуку помилок (debugging).

Формування ключових компетентностей:

- ✔ *Інформаційно-цифрова:* впевнене використання онлайн-сервісів для навчання.
- ✔ *Математична:* перетворення текстової умови задачі на математичну формулу.
- ✔ *Уміння вчитися впродовж життя:* здатність до самостійного аналізу помилок через автоматизований зворотний зв'язок.

Обладнання та наочність: персональні комп’ютери з доступом до Інтернету, мультимедійний проектор, презентація.

Програмне забезпечення: браузер (Chrome/Edge), середовище розробки IDLE Python (або онлайн-компілятор, вбудований в Eolymp).

Методичне забезпечення:

– інструкція з реєстрації на Eolymp;

– картки з умовами задач «Борги» та «Біткопійки»;

– презентація;

– Руденко В. Д. Інформатика (профільний рівень) : підруч. для 10 кл. закл. загал. серед. освіти / В. Д. Руденко, Н. В. Речич, В. О. Потієнко. — Харків : Вид-во «Ранок», 2019. — 256 с. : іл.

Хід уроку

І. Організаційний етап (2 хв)

- Привітання, перевірка присутніх.
- Перевірка готовності робочих місць (чи є доступ до Інтернету).
- Інструктаж з техніки безпеки в комп’ютерному класі.

ІІ. Мотивація навчальної діяльності (3 хв)

Слово вчителя: *Уявіть, що ви написали програму, а її перевіряє не вчитель, а робот – миттєво і безжально. Саме так працює олімпіадне програмування. Це спорт, де ваша зброя – код, а суперник – час та складність алгоритму. Сьогодні ми зробимо перший крок: навчимося «спілкуватися» з цим роботом на платформі Eolymp та розв'яжемо перші задачі, які були на реальних олімпіадах минулих років*.

ІІІ. Актуалізація опорних знань (3 хв)

· **Фронтальне опитування:**

1. Які арифметичні дії ви знаєте в Python? *(+, -, *, /)*.
2. Чим відрізняється ділення / від //? *(Дробове та цілочисельне)*.
3. Як ввести дані з клавіатури? *(Функція input())*.
4. Що таке змінна? *(Комірка пам'яті з іменем)*.

ІV. Вивчення нового матеріалу (12 хв)

(Демонстрація екрану вчителем)

Розпочнемо з головної відмінності олімпіадного програмування від того, що ми звикли бачити на звичайних уроках. Коли ви створюєте програму в школі, ви зазвичай пишете діалог: комп’ютер запитує «Введіть число», користувач вводить, а комп’ютер відповідає «Ваш результат дорівнює...». На

олімпіаді, зокрема на платформі Eolymp, вашим користувачем є не людина, а автоматична тестуюча система — робот.

Уявіть свою програму як «чорну скриньку». Робот "закидає" в неї вхідні дані (числа, рядки) і чекає на виході лише "сухий" результат. Цей робот не розуміє ввічливості. Якщо у відповідь на задачу «Додати два числа» ваша програма виведе фразу *"Відповідь: 10"*, робот порівняє це з еталоном, де написано просто *"10"*. Він побачить розбіжність у символах і зарахує це як помилку. Тому наше перше правило: **ніякого зайвого тексту у введенні та виведенні**. Програма має бути мовчазним обчислювальним інструментом.

Давайте поглянемо, як це реалізується технічно. У підручниках з інформатики для 10 класу (наприклад, авторства Руденка) ви часто бачите конструкцію: k = int(input("Через скільки днів?")). Це правильно для створення програм для людей, щоб вони розуміли, що вводити .

Однак для системи Eolymp це груба помилка. Текст у дужках функції input() буде сприйнятий системою як частина вашого виводу, і ви отримаєте помилку. Тому в олімпіадних задачах ми завжди залишаємо дужки порожніми: k = int(input()).

Також важливо розуміти типи даних. Функція input() зчитує все як рядок тексту. Щоб виконувати математичні дії, ми повинні явно перетворити цей текст на число. Якщо в задачі йдеться про кількість монет або грошей (як у задачі «Борги»), ми використовуємо цілі числа — int(). Якщо ж мова йде про координати чи відстані (як у задачі «Координати»), де можливі дробові значення, ми використовуємо дійсні числа — float().

Для розв’язання задач нам знадобиться більше, ніж просто додавання та множення. У Python є специфічні оператори, які є незамінними для олімпіад (вони наведені у таблицях вашого підручника):

- **Ділення:** Звичайний скіс / завжди дає дробове число (навіть якщо 4 поділити на 2, вийде 2.0). Але часто нам потрібне ціле число.

- **Цілочисельне ділення (//):** цей оператор відкидає дробову частину. Наприклад, 123 // 10 дасть нам 12. Це ідеальний спосіб «відрізати» останню цифру числа .
- **Остача від ділення (%):** це, мабуть, найважливіший оператор для початківців. Він показує те, що залишилося від ділення націло. Наприклад, 123 % 10 поверне 3 — останню цифру числа. Також за допомогою % 2 ми миттєво визначаємо, парне число чи ні.
- **Піднесення до степеня (**):** замість множення числа самого на себе багато разів, ми пишемо 2 ** 100. Важлива перевага Python у тому, що він, на відміну від інших мов (як C++ чи Pascal), вміє працювати з надзвичайно великими числами автоматично, не видаючи помилок переповнення пам’яті.

Коли ви відправите задачу на перевірку, система миттєво видасть вердикт. Ви повинні знати три основні статуси:

1. **AC (Accepted)** – горить зеленим. Це означає «Прийнято». Ваша програма пройшла всі тесті успішно. Це наша мета.
2. **WA (Wrong Answer)** – горить червоним. Це «Неправильна відповідь». Це означає, що програма запустилася, але видала результат, який не збігається з правильним. Найчастіші причини: помилка у формулі або ви забули прибрати зайвий текст типу «Result is:».
3. **CE (Compilation Error)** – помилка компіляції. Це означає, що ви допустили синтаксичну помилку (забули дужку, двокрапку або неправильно зробили відступ), і програма навіть не змогла запуститися.

Сьогодні на практичній частині ми закріпимо це: напишемо прості формули для олімпіадних задач «Борги» та «Прямокутник», використовуючи лише «сухе» введення та виведення, щоб отримати ваші перші зелені "AC".

V. Фізкультхвилинка

VI. Усвідомлення набутих знань та формування вмінь і навичок (20 хв)

(Робота на сайті Eolymp)

Попередня реєстрація учнів.

Приклад 1. Розв'язання задачі «Борги» (Рівень А)

Етап 1. Ознайомлення з умовою задачі.

Уважно читаємо умову: *в Аліси є початкова сума грошей **a**. Двоє людей (Боб і Світлана) повертають їй борги **b** і **c**, що збільшує її капітал. Водночас сама Аліса повинна віддати борг Дмитру **d**, що зменшує її капітал*.

Очікуваний результат: *кінцева сума грошей в Аліси*.

Етап 2. Аналіз задачі

- **Вхідні дані:** Чотири цілі числа (a, b, c, d), кожне з нового рядка. Обмеження: числа від 1 до 100.
- **Вихідні дані:** Одне ціле число.
- **Тип даних:** Оскільки всі числа цілі і результат невід'ємний, достатньо типу int (цілі числа).
- **Особливості:** Операції відбуваються послідовно, але порядок додавання/віднімання не впливає на результат (від перестановки доданків сума не змінюється).

Етап 3. Розробка алгоритму

Задача зводиться до лінійного алгоритму. Побудуємо математичну модель:

1. Початкова сума: **+a**.
2. Боб віддає борг: **+b**.
3. Світлана віддає борг: **+c**
4. Аліса віддає борг: **-d**.

Формула: **Result = a + b + c - d**.

Етап 4. Кодування

Реалізуємо алгоритм мовою Python. Використовуємо функцію input() для зчитування рядків та int() для перетворення їх у числа.

```
a = int(input())
```

```
b = int(input())
c = int(input())
d = int(input())
final_amount = a + b + c - d
# Виводимо результат
print(final_amount)
```

Етап 5. Тестування та налагодження

Етап 6. Оформлення розв'язку

Перевіряємо вимоги системи Eolymp:

1. Чи немає зайвого тексту в input() (наприклад, "Введіть a=")? Немає.
2. Чи немає зайвого тексту в print() (наприклад, "Відповідь:")? Немає.
3. Код чистий, готовий до відправки.

Приклад 2. Розв'язання задачі «Біткопійки» (Рівень А)

Етап 1. Ознайомлення з умовою задачі

В умові йдеться про Міс М, яка має призовий фонд у розмірі **n** біткопійок. Є **m** переможців. Кожному з них вона повинна виплатити фіксовану нагороду – 3 біткопійки.

Очікуваний результат: *скільки біткопійок залишиться у Міс М після роздачі призів.*

Етап 2. Аналіз задачі

- **Вхідні дані:** два цілі числа: **n** (загальна сума) та **m** (кількість людей). Зазвичай у таких задачах рівня А вони подаються в окремих рядках.
- **Вихідні дані:** одне ціле число – залишок.
- **Обмеження:** гарантується, що грошей вистачить **n >= 3 * m**

Етап 3. Розробка алгоритму

1. Визначаємо загальну суму виплат. Оскільки кожен з m переможців отримує 3 монети, то загальна виплата становить **3* m**.
2. Визначаємо залишок. Від початкової суми n віднімаємо суму виплат.

Математична модель: **Result = n - (3 * m).**

Етап 4. Кодування

Записуємо рішення мовою Python:

```
n = int(input())
m = int(input())
ans = n - m*3 # TODO
print(ans)
```

Етап 5. Тестування та налагодження

Етап 6. Оформлення розв'язку

Переконуємося, що формат введення/виведення відповідає стандартам олімпіадної системи (стандартні потоки введення-виведення, відсутність зайвих коментарів у виводі). Код готовий до завантаження в систему Eolymp.

Задача 3 (Додаткова). «Тура» (Рівень А)

Для тих, хто впорався швидше. Формула: n + m - 2.

VII. Підведення підсумків уроку (3 хв)

Рефлексія:

- Скільки «зелених галочок» (AC) ви отримали сьогодні?
- Яка найчастіша помилка була?

Оцінювання: формувальне оцінювання, відзначення лідерів.

VIII. Домашнє завдання (1 хв)

– **Завершити реєстрацію** на Eolymp (хто не встиг).

– **Дорішати задачі** уроку в режимі «Upsolving» (якщо не здали на 100%).

– **Задача наперед:** подумати над задачею «Середнє арифметичне» (знайти невідоме число, якщо відоме одне число і середнє арифметичне).

Додаток Г

План-конспект уроку № 2

«_____» _____ 20___ року

Урок № 2.

Тема: Цілочисельна арифметика

Мета:

- ✔ **навчальна:** сформувати поняття про цілочисельні типи даних та операції над ними в Python (//, %); навчити застосовувати ці операції для розв’язання задач на виділення цифр числа та роботу з координатами; закріпити навички написання лінійних алгоритмів.
- ✔ **розвиваюча:** розвивати логічне мислення (вміння розбивати число на розряди), математичну компетентність (виведення формул), пам'ять та увагу.
- ✔ **виховна:** виховувати самостійність, акуратність у написанні коду, вміння працювати на результат в умовах обмеженого часу.

Формування ключових компетентностей:

- ✔ *Математична:* перетворення текстової умови задачі у формулу;
- ✔ *Інформаційно-цифрова:* використання інструментів налагодження коду.

Обладнання та наочність: персональні комп’ютери з доступом до Інтернету, мультимедійний проектор, презентація.

Програмне забезпечення: браузер (Chrome/Edge), середовище розробки IDLE Python (або онлайн-компілятор, вбудований в Eolymp).

Методичне забезпечення: картки з формулами скороченого множення, роздруківка умов задач «Тура» та «Середнє арифметичне».

Тип уроку: комбінований.

Хід уроку

І. Організаційний етап (2 хв)

- Привітання.
- Перевірка присутності та готовності до уроку (логін в Eolymp).

II. Мотивація навчальної діяльності (3 хв)

Вчитель: *«Уявіть, що ви програмуєте банкомат. Клієнт хоче зняти 1250 гривень, а у вас є тільки купюри по 100. Як банкомат дізнається, скільки купюр видати (12) і скільки залишиться на рахунку або не видасться (50)? Звичайна математика тут працює повільно. У програмуванні для цього є супер-інструменти: ділення націло та остача. Сьогодні ми навчимося «розбирати» числа на частинки, як конструктор LEGO».*

III. Актуалізація опорних знань (3 хв)

Бліц-опитування (повторення):

1. Якою функцією ми зчитуємо дані? *(input()).*
2. Чому input() недостатньо для математики? *(Бо він зчитує текст, треба int()).*
3. Який вердикт системи означає «Все вірно»? *(**AC** – Зелений колір).*
4. Який вердикт означає «Помилка у відповіді»? *(**WA** – Червоний колір).*

IV. Вивчення нового матеріалу (10 хв)

(Демонстрація екрану вчителем)

Цілочисельне ділення (//). Це операція, яка відповідає на питання *«Скільки цілих разів число A вміщується в число B?».* Наприклад, 10 // 3 = 3 (бо 3 рази по 3 дає 9). Дробова частина відкидається. Часто використовується у завданнях де потрібно відкинути останню цифру числа: 123 // 10 → 12.

Остача від ділення (%). Це те, що «не помістилося» при діленні націло. Наприклад: 10 % 3 = 1 (бо 10 – 3 * 3 = 1). Часто використовується у завданнях де потрібно отримати останню цифру числа (взяти остачу від ділення на 10): 123 % 10 → 3. Також можна перевірити парність деякого числа: x % 2. Якщо результат 0 – парне, 1 – непарне.

Демонстрація фрагментів коду вчителем

V. Фізкультхвилинка

VI. Усвідомлення набутих знань та формування вмінь і навичок (22 хв)

(Робота в системі Eolymp)

Задача 1. «Середнє арифметичне» (Рівень В)

Етап 1. Ознайомлення з умовою задачі

У Петрика є два числа **a** та **b** (причому **a <= b**). Він обчислив їхнє середнє арифметичне **c**. Нам відомі число **a** (менше) та саме середнє арифметичне **c**. Потрібно знайти друге число – **b.**

Етап 2. Аналіз задачі

Вхідні дані: два цілі числа a та c. За умовою $a \le c \le 100$

Вихідні дані: ціле число b**.**

Тип даних: оскільки всі числа цілі, нам достатньо типу **int**.

Етап 3. Розробка алгоритму

Математична модель задачі (згадаємо формулу середнього арифметичного):

$$c = \frac{a+b}{2} \rightarrow 2 * c = a + b \rightarrow b = 2 * c - a$$

Етап 4. Кодування (самостійно учнями)

```
a = int(input())
c = int(input())
ans = 2*c - a # TODO
print(ans)
```

Типова помилка: учні намагаються використати ділення. Вчитель нагадує: *Уникайте ділення (/), бо воно створює дробові числа (float), які можуть бути неточними. Використовуйте множення*.

Етап 5. Тестування та налагодження

Етап 6. Оформлення розв'язку

Задача 2. «Тура» (Рівень А)

Етап 1. Ознайомлення з умовою задачі

На шахівниці розміром **n×m** (де **n** – кількість рядків, **m** – кількість стовпчиків) у нижньому лівому куті стоїть тура. Потрібно знайти кількість клітинок, на які тура може перемістится за один хід. Тура ходить тільки по вертикалі та горизонталі 3.

<u>Етап 2.</u> Аналіз задачі

- **Вхідні дані:** два натуральних числа n і m (до 20).
- **Вихідні дані:** одне ціле число – кількість доступних клітинок.
- **Візуалізація**: уявіть туру в кутку. Вона «бачить» усі клітинки перед собою вгору і всі клітинки праворуч.

<u>Етап 3.</u> Розробка алгоритму

Не потрібно моделювати рух фігури чи створювати масив. Це задача на просту логіку.

1. *Скільки клітинок у стовпчику (по вертикалі)?* Всього n. Тура стоїть на одній з них. Отже, вільних для ходу: n - 1.
2. *Скільки клітинок у рядку (по горизонталі)?* Всього m. Тура займає одну. Вільних для ходу: m - 1.
3. *Загальна кількість ходів* – це сума ходів по вертикалі та горизонталі.

Формула: ans = (n - 1) + (m - 1) або спрощено n + m - 2.

Етап 4. Кодування

```
n = int(input())
m = int(input())
ans = n + m - 2 # TODO
print(ans)
```

<u>Етап 5.</u> Тестування та налагодження

<u>Етап 6.</u> Оформлення розв'язку

Задача 3 (Додаткова). «Цифри числа»

Якщо є в Eolymp, або аналог. Знайти суму цифр двозначного числа.

Python

```
n = int(input())
s = (n // 10) + (n % 10)
print(s)
```

VII. Підведення підсумків уроку (3 хв)

Рефлексія:

- Скільки «зелених галочок» (AC) ви отримали сьогодні?
- Яка найчастіша помилка була?

Фронтальне запитання:

У нас є число 57. Якою одною операцією отримати 5? *(// 10).* А якою 7? *(% 10).*

VIII. Домашнє завдання (1 хв)

- Закінчити задачі уроку в режимі Upsolving.
- Ø Поміркувати над задачею «Міс М та квіти» (B) (знадобиться на наступному уроці).

Додаток Д

План-конспект уроку № 3

«___»_____ 20___ року

Урок № 3.

Тема: Математичне моделювання та геометрія.

Мета:

- ✔ **навчальна:** ознайомити учнів з можливостями модуля math (функції sqrt, pow, константа pi); навчити реалізовувати формули обчислювальної геометрії (відстань між точками, площа фігур) мовою Python; сформувати навички розв’язання задач із дробовими числами (float).
- ✔ **розвиваюча:** розвивати просторову уяву, вміння аналізувати геометричні задачі, навички роботи зі стандартними бібліотеками.
- ✔ **виховна:** виховувати уважність до точності обчислень, культуру написання коду (імпорт модулів).

Формування ключових компетентностей:

- ✔ *Математична:* застосування теореми Піфагора та формул площ у прикладному контексті.
- ✔ *Інформаційно-цифрова:* використання бібліотек для розширення функціоналу мови.

Обладнання та наочність: персональні комп’ютери з доступом до Інтернету, мультимедійний проектор, презентація.

Програмне забезпечення: браузер (Chrome/Edge), середовище розробки IDLE Python (або онлайн-компілятор, вбудований в Eolymp).

Методичне забезпечення:

– Довідник функцій модуля math (роздруківка або файл),

– Умови задач «Координати» та «Прямокутник».

– Руденко В. Д. Інформатика (профільний рівень) : підруч. для 10 кл. закл. загал. серед. освіти / В. Д. Руденко, Н. В. Речич, В. О. Потієнко. — Харків : Вид-во «Ранок», 2019. — 256 с. : іл.

Тип уроку: комбінований

Хід уроку (45 хв)

І. Організаційний етап (2 хв)

- Привітання.
- Коротка перевірка домашнього завдання

ІІ. Мотивація навчальної діяльності (3 хв)

Вчитель: *«На попередніх уроках ми працювали з цілими числами (гроші, біткопійки). Але світ не складається лише з цілих чисел. У комп'ютерних іграх персонажі рухаються по координатах, ракети літають за траєкторіями, а архітектори рахують площі. Сьогодні ми навчимо Python бути справжнім інженером: рахувати корені, відстані та працювати з числом пі. Для цього ми підключимо нашу першу «суперсилу» — бібліотеку math*».

ІІІ. Актуалізація опорних знань (3 хв)

Фронтальне опитування:

1. Який тип даних відповідає за дробові числа? (float).
2. Як зчитати дробове число з клавіатури? (x = float(input())).
3. Як піднести число x до квадрата без спеціальних бібліотек? (x ** 2).

ІV. Вивчення нового матеріалу (10 хв)

(Демонстрація екрану вчителем. Матеріал базується на підручнику 10 кл., розділ 2.4)

Python має набори готових функцій. Щоб їх використати, треба написати: import math. **Важливо:** імпорти пишемо в самому першому рядку програми. До основних функцій модуля math відносять:

- **math.sqrt(x)** — квадратний корінь ($\sqrt{x}$).
- **math.fabs(x)** — модуль числа (|x|). *Примітка: для цілих чисел можна abs(x).
- **math.pi** — число pi (3.14159...).
- **math.ceil(x)** — округлення вгору (стеля).
- **math.floor(x)** — округлення вниз (підлога).

Важливо вміти перетворювати геометричні формули в програмний код. Наприклад, для знаходження гіпотенузи (**c**) трикутника, якщо відомо його катети a, b, слід написати геометричну формулу: $c = \sqrt{a^2 + b^2}$. Програмний код виглядатиме наступним чином**: c = math.sqrt(a**2 + b**2).**

Для сьогоднішньої практики давайте пригадаємо формулу **відстані між точками (2D та 3D):**

На площині: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

У просторі: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + (z_2 - z_1)^2$.

V. Фізкультхвилинка

VI. Усвідомлення набутих знань та формування вмінь і навичок (22 хв)

(Робота в системі Eolymp)

Задача 1. «Координати» (Рівень B)

<u>Етап 1.</u> Ознайомлення з умовою задачі

Дано точку з координатами (x, y, z) у тривимірному просторі. Необхідно знайти квадрат відстані від цієї точки до центру координат – точки (0, 0, 0).

<u>Етап 2.</u> Аналіз задачі

- **Вхідні дані:** Три цілі числа x, y, z. Діапазон: від -100 до 100.
- **Вихідні дані:** Одне число – квадрат відстані.
- **Тип даних:** Вхідні числа невеликі, але при піднесенні до квадрата і додаванні результат може зрости. Проте для 100^2 + 100^2 + 100^2 = 30000 стандартного цілого типу int вистачить із запасом.
- **Геометрична довідка:** Відстань у 3D-просторі обчислюється аналогічно теоремі Піфагора.

<u>Етап 3.</u> Розробка алгоритму (Математична модель)

Згадаємо формулу відстані між двома точками у просторі. Оскільки одна з точок – це початок координат (0, 0, 0), формула спрощується:

$$d = \sqrt{(x - 0)^2 + (y - 0)^2 + (z - 0)^2} = d = \sqrt{x^2 + y^2 + z^2}$$

Увага! В умові просять знайти квадрат відстані (d^2). Це означає, що корінь добувати не потрібно!

d^2 = x^2 + y^2 + z^2

Це значно спрощує задачу: нам не потрібен модуль math і функція sqrt, достатньо простого піднесення до степеня.

Етап 4. Кодування

Реалізуємо формулу:

```
x = int(input())
y = int(input())
z = int(input())
ans = x**2 + y**2 + z**2 # TODO
print(ans)
```

Етап 5. Тестування та налагодження

Етап 6. Оформлення розв'язку

Задача 2. «Міс М та квіти» (Рівень В)

Етап 1. Ознайомлення з умовою задачі

Є квіткова клумба розміром **n x m**. Робот-садівник стартує у лівому верхньому куті. Він може рухатися тільки вправо або вниз. Коли він доходить до правого нижнього кута, він повертається на старт. За перший прохід він збирає всі квіти на своєму шляху. За кожен наступний прохід він може зібрати лише по одній квітці (бо більшість вже зібрано). Питання: *за скільки проходів робот збере абсолютно всі квіти?*

Етап 2. Аналіз задачі

- **Вхідні дані:** Розміри клумби
- **Вихідні дані:** Кількість проходів.
- **Ключовий момент:** Загальна кількість квітів на клумбі – це площа прямокутника (**n * m**). Нам треба дізнатися, скільки квіток він збере за *перший* раз.

Етап 3. Розробка алгоритму

1. *Скільки клітинок відвідує робот за один прохід від (1,1) до (n, m)?*

Будь-який шлях, що веде тільки вправо і вниз у прямокутнику *n x m*, має однакову довжину. *Кількість кроків:* (n - 1) вниз + (m - 1) вправо. *Кількість відвіданих клітинок = Кількість кроків + 1 (стартова)* = **n + m - 1**. Отже, за 1-й прохід зібрано: K1 = n + m - 1.

2. *Скільки квітів залишилося?*

res = (n * m) - (n + m - 1).

Усі інші квіти він збирає по одній за прохід. Тобто кількість додаткових проходів дорівнює res.

3. *Загальна кількість проходів*:

total = 1 (перший) + Res = 1 + (n * m - n - m + 1). Спростимо формулу математично (групування):

total = nm - n - m + 2 = n(m-1) - (m-1) + 1 = (n-1)(m-1) + 1.

Фінальна формула: **ans = (n - 1) * (m - 1) + 1**.

<u>Етап 4.</u> Кодування

```
n = int(input())
m = int(input())
ans = (n - 1) * (m - 1) + 1 # TODO
print(ans)
```

<u>Етап 5.</u> Тестування та налагодження

<u>Етап 6.</u> Оформлення розв’язку

VIII. Домашнє завдання (1 хв)

Пройти тренувальне змагання на Eolymp за підсумками 1 тижня.