Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка

Фізико-математичний факультет
Кафедра інформатики та методики її навчання

Кваліфікаційна робота

МЕТОДИКА ПІДГОТОВКИ УЧНІВ ДО ОЛІМПІАДИ З ІНФОРМАТИКИ

Спеціальність 014 Середня освіта

Освітня програма «Інформатика, математика, STEM-освіта»

Здобувача другого (магістерського)
рівня вищої освіти
Бенеша Андрія Станіславовича

НАУКОВИЙ КЕРІВНИК:
кандидат фіз.-мат. наук, доцент
Мартинюк Сергій Володимирович

РЕЦЕНЗЕНТ:
завідувач кафедри інформаційних
технологій і програмування
Українського державного університету
імені Михайла Драгоманова,
кандидат пед. наук, доцент
Єфименко Василь Володимирович

Тернопіль — 2025

АНОТАЦІЯ

Бенеш А. С. Методика підготовки учнів до олімпіади з інформатики. Кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 014 Середня освіта. ТНПУ ім. В. Гнатюка. Тернопіль, 2025. 74 с.

У кваліфікаційній роботі розгляну методику підготовки учнів до олімпіад з інформатики. Здійснено аналіз наукових джерел, огляд тенденцій, класифікацію задач, спостереження, експеримент та анкетування. Теоретична частина описує історію олімпіад, еволюцію завдань, методичні аспекти підготовки здобувачів, а практична — форми підготовки, використання платформам (E-olymp, Codeforces, Timus, CSES, AtCoder), добір завдань, характеристики розв’язування задач, результати опитування. На основі отриманих даних розроблено методичні рекомендації щодо підготовки учнів до олімпіад з інформатики, що можуть бути впроваджені в освітній практиці.

Ключові слова: інформатика, олімпіада, методика, алгоритмічне мислення, програмування.

ABSTRACT

Benesh A. S. Methodology for training students for the Computer Science Olympiad. Master's thesis for the MA degree in the specialty 014 Secondary education. Ternopil Volodymyr Hnatiuk National Pedagogical University. Ternopil, 2025. 74 p.

This thesis examines methods for preparing students for computer science competitions. It includes an analysis of scientific sources, a review of trends, a classification of tasks, observations, experiments, and surveys. The theoretical part describes the history of competitions, the evolution of tasks, and methodological aspects of preparing applicants, while the practical part describes forms of preparation, the use of platforms (E-olymp, Codeforces, Timus, CSES, AtCoder), the selection of tasks, characteristics of problem solving, and survey results. Based on the data obtained, methodological recommendations for preparing students for computer science Olympiads have been developed, which can be implemented in educational practice.

Keywords: computer science, olympiad, methodology, algorithmic thinking, programming.

ЗМІСТ

ВСТУП

Олімпіадний рух у сучасній системі освіти посідає особливе місце, адже він не лише сприяє виявленню та підтримці обдарованої молоді, але й забезпечує формування в учнів навичок високого рівня, що виходять далеко за межі шкільної програми. Участь у змаганнях з інформатики розглядається сьогодні як один із найефективніших способів розвитку алгоритмічного мислення, творчих здібностей та вміння розв’язувати складні інтелектуальні завдання. В умовах стрімкого розвитку цифрових технологій та зростання ролі інформаційних систем саме інформатика стає тією дисципліною, де поєднання теоретичних знань і практичних навичок має першочергове значення.

Олімпіади з інформатики відображають рівень підготовки учнів у сфері алгоритмів, структур даних, програмування та аналізу складних проблемних ситуацій. Вони водночас є і майданчиком для перевірки особистих умінь, і стартовим майданчиком для майбутньої професійної діяльності в ІТ-галузі. Тому проблема ефективної підготовки школярів до участі в таких змаганнях набуває особливої ваги та потребує системного підходу, що поєднує навчальні, методичні та організаційні аспекти.

Актуальність теми зумовлена тим, що сучасний освітній процес часто не встигає за темпами розвитку інформаційних технологій. Шкільна програма з інформатики формує лише базові знання, тоді як олімпіадні завдання вимагають від учня вміння працювати з алгоритмами високого рівня, аналізувати великі обсяги даних, застосовувати оптимізовані методи програмування та працювати у швидкому темпі. Крім того, сучасні тенденції розвитку олімпіадного руху свідчать про зростання складності конкурсних завдань та посилення конкуренції, що ще більше актуалізує потребу в якісній підготовці учнів. Таким чином, розробка та впровадження ефективної методики підготовки школярів до олімпіади з інформатики є науково та практично значущим завданням.

Метою роботи є обґрунтування та розробка методики підготовки учнів до олімпіади з інформатики, яка поєднує навчально-організаційні та практико-

орієнтовані елементи й забезпечує підвищення результативності їх участі у змаганнях.

Для досягнення цієї мети поставлено **завдання**:

— проаналізувати сучасний стан і тенденції розвитку олімпіадного руху з інформатики;

— дослідити погляди науковців і методистів на проблему підготовки учнів;

— здійснити класифікацію типів олімпіадних завдань і методів їх розв’язання;

— охарактеризувати появу й розвиток напряму «Інформаційні технології»;

— розробити та апробувати авторський набір задач для тренування.

Об’єкт дослідження: процес підготовки учнів до участі в олімпіадах з інформатики.

Предмет дослідження: методичні підходи, форми та засоби організації навчання, спрямовані на формування в учнів необхідних компетентностей для успішної участі в інтелектуальних змаганнях.

У процесі дослідження застосовувалися такі **методи**: теоретичний аналіз науково-методичної літератури та сучасних освітніх програм; систематизація й узагальнення даних про організацію олімпіадного руху; педагогічне спостереження й анкетування учнів; статистичний аналіз результатів участі школярів у змаганнях; експериментальна перевірка ефективності розробленої методики через практичні заняття та порівняння показників до й після її впровадження.

Практичне значення роботи полягає у створенні комплексної методики підготовки учнів до олімпіад з інформатики, яка може використовуватися вчителями у процесі навчання на уроках, у гуртковій роботі та під час організації спецкурсів. Запропоновані авторські добірки задач, рекомендації щодо використання онлайн-платформ, модельний план підготовки та система оцінки результатів здатні стати основою для розробки програм позакласної та

факультативної роботи. Вони сприятимуть підвищенню ефективності навчання, формуванню алгоритмічного мислення учнів і розвитку їхнього інтелектуального потенціалу.

Структура роботи: робота складається зі вступу, двох розділів теоретичного й практичного характеру, третього розділу, що містить опис авторської методики, висновків та списку використаних джерел. У першому розділі розглядаються сучасні тенденції олімпіадного руху та погляди науковців і методистів на проблему підготовки. Другий розділ присвячений практичним аспектам: організаційним формам навчання, використанню онлайн-платформ, розробці та аналізу авторського набору задач, а також оцінці ефективності методики. У третьому розділі представлено розробку власної невеликої програми підготовки учнів до олімпіад з інформатики. Загальний обсяг роботи – 74 сторінки.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ПІДГОТОВКИ УЧНІВ ДО ОЛІМПІАД З ІНФОРМАТИКИ

1.1. Стан і сучасні тенденції розвитку олімпіадного руху з інформатики

Початки олімпіадного руху з інформатики тісно пов’язані з розвитком самої інформатики як науки та навчальної дисципліни. У світі перші кроки в цьому напрямку почалися у 1970-х роках, коли програмування стало окремим предметом для вивчення у школах, а комп’ютери почали з’являтися в освітніх установах. Згодом виникла потреба у змаганнях між учнями, які не лише вивчали основи комп’ютерної грамотності, а й демонстрували глибоке розуміння алгоритмів, логіки та вміння розв’язувати складні задачі за допомогою програмування [1].

Офіційним початком міжнародного змагання вважається проведення першої Міжнародної олімпіади з інформатики (IOI) у 1989 році в Болгарії. Цей конкурс започаткувала ЮНЕСКО як ініціативу підтримки обдарованих учнів у сфері ІТ, і з того часу IOI стала найпрестижнішим світовим змаганням з алгоритмічного програмування для школярів. Формат олімпіади передбачає індивідуальне розв’язання складних алгоритмічних задач у комп’ютерному середовищі за обмежений час, що і сьогодні залишається стандартом.

В Україні перші олімпіади з інформатики почали проводитись ще в 1980-х роках. Українські школярі брали участь у Всесоюзних олімпіадах, а з 1992 року, після здобуття незалежності, наша країна сформувала власну систему шкільних олімпіад з інформатики. Уже з перших років незалежності Україна активно долучилася до міжнародного олімпіадного руху: починаючи з 1994 року українські учні стали постійними учасниками IOI [6, 24, с. 134-138].

Протягом останніх десятиліть в Україні сформувалася багаторівнева система проведення олімпіад: від шкільного етапу до всеукраїнського. Такий підхід дозволяє виявити та підтримати здібних учнів на ранніх етапах навчання. Олімпіади з інформатики набули особливої популярності у 2000-х роках, коли

доступ до комп’ютерної техніки значно покращився, а кількість учнів, зацікавлених у програмуванні, почала стрімко зростати.

Значну роль у розвитку цього напрямку відіграли українські науковці, методисти, вчителі-ентузіасти, які впроваджували нові підходи до підготовки учнів, адаптували зарубіжні методики та створювали власні. Паралельно розвивалася і технічна база: з’явилися спеціалізовані навчальні платформи, середовища програмування, навчальні посібники українською мовою.

Олімпіадний рух в Україні відзначається стабільністю та результативністю: українські школярі регулярно виборюють медалі на міжнародних олімпіадах. Це свідчить про високий рівень національної підготовки, яка поєднує глибокі теоретичні знання з постійною практикою розв’язування задач [2].

Останні роки характеризуються розширенням поняття «олімпіада з інформатики». Поряд з традиційними олімпіадами з програмування почали проводитися конкурси з інформаційних технологій, хакатони, турніри з розробки програмного забезпечення. Це зумовлено розвитком цифрової галузі та зростанням попиту на різні ІТ-компетентності [22, с. 44-47].

Розвиток олімпіадного руху з інформатики зумовив поступову еволюцію як організаційних форм проведення змагань, так і самих завдань, які пропонують учням. Якщо на початкових етапах основна увага приділялася перевірці базових навичок програмування й умінню реалізовувати прості алгоритми, то з часом зміст олімпіадних завдань зазнав суттєвих змін, ставши значно глибшим, складнішим і ближчим до академічного та інженерного рівня.

На ранньому етапі задачі здебільшого мали прикладний характер: учням пропонувалося написати програму, що обробляє прості текстові або числові дані, виконує базові обчислення чи моделює елементарні процеси. Такі завдання сприяли формуванню у школярів навичок синтаксично правильного програмування, базового розуміння циклів, умов та структур даних. Часто завдання були вузько орієнтовані на конкретну мову програмування

(здебільшого Pascal або Basic), а складність полягала, радше, у реалізації технічних вимог, ніж в алгоритмічному мисленні.

У 1990-х роках, з розвитком алгоритмічної бази і популяризацією мови програмування C/C++, у завданнях почали з’являтися класичні задачі з комбінаторики, теорії чисел, пошуку в графах, динамічного програмування, які вимагали не лише вміння програмувати, але й ґрунтовного знання алгоритмів. Така зміна орієнтації зробила олімпіади з інформатики ближчими до академічної дисципліни «Алгоритми та структури даних», що викладається в університетах.

Починаючи з 2000-х років завдання стають дедалі більш абстрактними, у них все рідше трапляються реальні життєві сюжети, а дедалі частіше — математичні моделі, що потребують побудови ефективних, оптимізованих рішень. Також відбувся перехід від тестування лише кінцевого результату до повноцінної автоматизованої перевірки коректності, ефективності та стабільності розв’язку у великому наборі тестів. Це дозволило суттєво підвищити якість змагань і рівень чесності в оцінюванні [20].

Одночасно, змінилися й вимоги до технічної підготовки учасників. Сучасні задачі, окрім знання класичних алгоритмів, передбачають перевірку вміння працювати з обмеженнями по часу та пам’яті, ефективно використовувати стандартні бібліотеки мов програмування, застосовувати нестандартні прийоми оптимізації. Деякі задачі вимагають навіть елементів евристики, розуміння машинної арифметики, обробки випадкових даних.

Також важливо відзначити появу нових форматів задач, що виникли у відповідь на розвиток змагального програмування в онлайн-середовищах. Наприклад, завдання типу «interaction» (взаємодія з середовищем), «approximation» (пошук неідеального, але доброго розв’язку) чи «output-only» (завдання, де потрібно лише згенерувати правильну відповідь на основі великого обсягу вхідних даних).

З появою нових технологічних рішень, мов програмування, парадигм розробки програмного забезпечення й підходів до обробки даних змінився і підхід до формування конкурсних завдань [14].

Зростання популярності багатопотокового програмування, робота з API, обробка потоків даних в реальному часі, концепції «інтернету речей», машинного навчання — все це опосередковано впливає на зміст і формат завдань, з якими стикаються учасники олімпіад. Звичайно, далеко не всі ці технології безпосередньо інтегруються в шкільний формат змагань, проте концептуально вони задають вектор розвитку: від суто математичних моделей до прикладних, гнучких, наближених до реального програмування.

Крім того, змінюється і рівень вимог до оптимізації. Сучасна ІТ-сфера потребує не просто «робочого коду», а рішень, здатних працювати стабільно в умовах великого навантаження. Тому задачі часто містять обмеження, які змушують учасника шукати неочевидні, більш витончені алгоритмічні підходи.

Завдяки цьому олімпіади сьогодні стали не лише майданчиком для перевірки знань, а й важливим інструментом формування конкурентоздатних навичок, які відповідають реаліям сучасного ІТ-світу. Саме тому вплив ІТ-галузі на зміст конкурсних завдань не можна розглядати як зовнішній фактор — він є внутрішньою рушійною силою, що змінює саму суть і спрямування олімпіад з інформатики [2].

Зважаючи на тісний зв’язок змісту олімпіадних завдань із реаліями ІТ-сфери, природним стало і розширення самого поняття «олімпіада з інформатики». Якщо донедавна ці змагання асоціювалися виключно з алгоритмічним програмуванням, то з розвитком цифрової освіти та змін у шкільних навчальних програмах з’явилися нові напрями, які відображають ширший спектр ІТ-компетентностей. Насамперед ідеться про розмежування між класичними олімпіадами з інформатики (які залишаються орієнтованими на програмування та алгоритмічне мислення) та конкурсами з інформаційних технологій, що мають іншу методичну і дидактичну основу.

Олімпіади з інформатики в традиційному розумінні зосереджуються на розв’язуванні складних алгоритмічних задач у мовах програмування високого рівня — таких як C++, Python, іноді Java. Ці змагання вимагають від учасників ґрунтовної підготовки в галузі алгоритмів, структур даних, теорії графів, динамічного програмування тощо. Основний акцент тут робиться на логіці, оптимізації, глибокому розумінні теоретичних основ комп’ютерної науки.

Натомість конкурси з інформаційних технологій орієнтуються переважно на прикладні аспекти: роботу з офісними пакетами, базами даних, створення презентацій, обробку графіки, розробку вебсайтів, основи кібербезпеки та навіть 3D-моделювання. Такий формат змагань передбачає інше коло компетентностей — здатність ефективно використовувати готові програмні засоби, інтегрувати інструменти та вирішувати практичні завдання, подібні до тих, що виникають у сучасному цифровому середовищі.

В Україні поділ між цими двома напрямами почав оформлюватися приблизно десять років тому, коли окремо почали проводитися змагання з інформаційних технологій. Це дозволило залучити до участі ширше коло учнів, які цікавляться не лише алгоритмічним програмуванням, а й іншими ІТ-напрямами. Такий підхід сприяв більшій диференціації обдарованої молоді та розвитку освітніх траєкторій відповідно до реальних уподобань і здібностей учнів [4].

Сьогодні обидва напрями існують паралельно й доповнюють один одного: олімпіади з інформатики формують фундаментальні алгоритмічні навички, а конкурси з інформаційних технологій — практичну ІТ-грамотність.

Отже, сучасний олімпіадний рух не лише розширив свою аудиторію, а й набув гнучкості, відкривши можливості для кожного учня знайти свій власний шлях у світі цифрової освіти.

1.2. Погляди науковців і методистів на олімпіадну підготовку з інформатики

Методична література, наукові статті та практичні посібники, що з’явилися протягом останніх десятиліть, відображають еволюцію підходів до формування алгоритмічної компетентності, структурування процесу підготовки та організації ефективної роботи з обдарованими учнями.

Серед українських дослідників особливої уваги заслуговують праці Миколи Морзе, який у своїх роботах акцентує на компетентнісному підході до навчання інформатики в школі, підкреслюючи значення розвитку алгоритмічного мислення через системну роботу з задачами. Крім того, у працях Світлани Литвин, присвячених моделюванню навчального середовища для розвитку творчих здібностей учнів, простежується ідея поєднання класичної підготовки з інноваційними методами, зокрема з використанням сучасних цифрових платформ для самостійної роботи учнів.

Методисти, як-от Олена Власенко та Лариса Бондаренко, наголошують на важливості поетапної підготовки до олімпіад, яка включає формування базових навичок програмування, поступовий перехід до складніших алгоритмічних тем та обов’язкове розв’язування типових задач за темами. У методичних рекомендаціях, зокрема в матеріалах ІМЗО (Інституту модернізації змісту освіти) також звертається увага на потребу забезпечення рівного доступу до якісної підготовки, включаючи використання відкритих онлайн-ресурсів.

Західноєвропейські автори, серед яких варто згадати, наприклад, Хайнца-Гюнтера Пейєра — одного з методистів, залучених до організації європейських олімпіад з інформатики, підкреслюють важливість міждисциплінарного підходу до задач, де програмування інтегрується з елементами комбінаторики, дискретної математики, теорії інформації. У його дослідженнях простежується тенденція до зменшення «штучної» складності задач на користь тієї, що відображає реальні виклики ІТ-галузі, з особливим акцентом на формування гнучкого мислення [27].

Іншим важливим джерелом сучасної методики є щорічні збірники задач і аналізів з міжнародних олімпіад, таких як IOI (International Olympiad in Informatics), де публікуються не лише самі завдання, а й детальні розбори варіантів розв'язків. Це дозволяє педагогам формувати ефективні тренувальні моделі, орієнтовані на міжнародні стандарти. У сучасній методичній літературі також посилюється акцент на самостійну дослідницьку діяльність учнів. Наприклад, у публікаціях Центру обдарованої молоді (Gifted Education Center) пропонується підхід, де учень виступає не лише виконавцем, а й дослідником, котрий аналізує структуру задачі, експериментує з різними підходами до її розв'язання, вчиться оцінювати ефективність коду.

Важливо звернутися до аналізу конкретних методичних рекомендацій та авторських розробок, що безпосередньо впливають на практику підготовки школярів до олімпіад з інформатики. Саме ці напрацювання стають основою для формування навчальних програм, проведення гурткової роботи та організації ефективного освітнього процесу в умовах позакласної підготовки.

В Україні важливу роль у методичному супроводі цього напрямку відіграє Інститут модернізації змісту освіти (ІМЗО), який регулярно публікує методичні рекомендації щодо викладання інформатики та організації роботи з обдарованими учнями. У відповідних рекомендаціях фахівці акцентують увагу на необхідності поетапної підготовки до олімпіад, яка передбачає: освоєння базових алгоритмічних структур, системне розв'язування задач підвищеної складності, участь у тренуваннях на онлайн-платформах та розвиток навичок аналізу коду. Особлива увага приділяється необхідності інтегрувати позакласну підготовку в загальну систему роботи з учнем, з урахуванням його мотивації, індивідуального стилю навчання та можливостей закладу освіти.

У контексті практичних розробок варто згадати посібник «Олімпіадна Інформатика» (2024), створений за ініціативи Малої академії наук України у співпраці з учасниками українських команд на міжнародних змаганнях [15]. У ньому представлено структуровану добірку алгоритмічних тем, короткі теоретичні викладки та систематизовані задачі різного рівня складності. Автори

посібника (зокрема, Андрій Андрєєв, член журі обласних етапів олімпіад, та інші практики) орієнтуються на наближення умов до реального формату IOI, що робить матеріали цінними для викладачів і тренерів гуртків.

Також, заслуговує на увагу збірник «Задачі з програмування 2018» під загальною редакцією Світлани Ярославівни Ковальової — одного з координаторів проведення онлайн-змагань [16]. В матеріалах подано детальні аналізи типових помилок учнів, алгоритмічні підказки та приклади оптимізованих розв’язків, що є безцінним джерелом для викладачів, які хочуть збудувати ефективну програму підготовки.

Серед зарубіжних авторських розробок особливо впливовими залишаються роботи фінського викладача Антті Лааксонена, зокрема його відомий посібник «Competitive Programmer’s Handbook», який широко використовується у тренуваннях до олімпіад [17]. Ця праця не лише охоплює ключові алгоритмічні теми, а й дає практичні поради щодо стратегії розв’язування задач, управління часом на змаганнях та перевірки ефективності рішень. Посібник вільно доступний англійською мовою, що робить його популярним серед викладачів і учнів в усьому світі, включно з Україною.

Практика олімпіадної підготовки демонструє, що найбільший ефект спостерігається за умов системної роботи в рамках гуртків або спецкурсів, що проводяться за межами основної шкільної програми. Тут учні мають можливість поглиблено вивчати алгоритми, працювати над нестандартними задачами та отримувати зворотний зв’язок від досвідчених наставників. Важливою перевагою таких форматів є можливість адаптації темпів та змісту занять до рівня групи або конкретного учня, що практично неможливо реалізувати в умовах звичайного уроку [3].

Дослідження, проведені на базі закладів із профільним спрямуванням або участю у всеукраїнських проєктах підтримки обдарованих учнів, показують: систематична гурткова робота (мінімум два рази на тиждень) із чітко структурованим планом тем і постійним розв’язуванням задач підвищеної складності значно підвищує ймовірність успішного виступу на обласному та

всеукраїнському рівнях. При цьому окремо підкреслюється важливість змагань тренувального типу, у яких учень вчиться працювати в часовому обмеженні, проходить реальні тести та порівнює свої результати з іншими.

Разом із тим не менш важливим чинником визнано й роль сучасних технологій. Онлайн-платформи, такі як E-Olymp, Codeforces, CSES, LeetCode, дозволяють забезпечити регулярність тренувань і доступ до величезної кількості задач різного рівня. Ці ресурси також дають можливість учням самостійно обирати траєкторію навчання, розв’язувати задачі в будь-який зручний час і відслідковувати свій прогрес. За умови правильного педагогічного супроводу вони можуть стати повноцінною формою підготовки, особливо для учнів із віддалених, або малокомплектних шкіл, де немає можливості створити повноцінний гурток.

Узагальнюючи, можна стверджувати, що ефективність підготовки до олімпіад значною мірою залежить від поєднання кількох чинників: постійної практики з розв’язування задач, компетентного педагогічного супроводу, змагального досвіду та доступу до сучасних освітніх платформ. Жодна окрема форма не може забезпечити повноцінного результату без взаємодії з іншими. Саме тому комплексний підхід, який поєднує очні заняття, онлайн-інструменти, індивідуальні консультації та участь у змаганнях різного рівня, розглядається як найбільш продуктивна модель сучасної олімпіадної підготовки.

1.3. Характеристика основних типів олімпіадних завдань і методів їх розв’язання

Розглянемо основні типи завдань, які можуть бути запропоновані учасникам під час змагань. Різноманітність задач вимагає від учнів не лише володіння певними програмними інструментами, а й здатності до глибокого аналітичного мислення, розв’язування складних алгоритмічних проблем. Кожен тип задач вимагає застосування специфічних методів розв’язання, що дозволяє навчати учнів різним технікам програмування і алгоритмам [22].

Одним з найважливіших етапів у підготовці до олімпіад є вивчення різних типів задач, а також їхня класифікація, оскільки це дозволяє систематизувати знання й ефективно готувати учнів до конкретних вимог змагань. Зазвичай задачі поділяють на кілька основних категорій [15].

1. Алгоритмічні задачі

Алгоритмічні задачі є основою будь-якої олімпіади з інформатики, оскільки вони вимагають від учасників розв'язування проблем за допомогою програмування. Вони можуть охоплювати такі теми, як розрахунок числових значень, пошук найкоротших шляхів, пошук мінімуму або максимуму у масивах даних тощо. Алгоритмічні задачі часто використовують класичні алгоритми пошуку, сортування, розбиття масивів або побудови структур даних, таких як стек, черга, дерево, хеш-таблиця тощо. Задачі цієї категорії вимагають точного і швидкого вирішення за допомогою ефективних алгоритмів, що здатні обробляти великі обсяги даних.

Типові методи розв'язання включають:

— розбиття задачі на підзадачі (де кожну частину можна вирішити за допомогою окремого алгоритму);
— використання пошукових алгоритмів (наприклад, пошук в глибину, пошук в ширину);
— рекурсивні алгоритми для розв'язування задач типу «розділяй і володарюй».

2. Комбінаторні задачі

Комбінаторні задачі вимагають знаходження всіх можливих варіантів розв'язку або оптимальних рішень серед великої кількості варіантів. Вони часто пов'язані з проблемами на перестановки, розташування об'єктів, комбінації, вибір підмножин. Такі задачі часто зустрічаються в різноманітних турнірах з програмування та вимагають застосування методів перебору, генерації всіх варіантів та обчислення кількості можливих результатів.

Типові методи розв’язання включають:

— метод перебору (генерація всіх можливих варіантів і відбір правильних);
— динамічне програмування для обчислення оптимальних шляхів у комбінаторних задачах;
— методи на основі теорії графів для розв’язування задач типу «найкоротший шлях» або «максимальний потік».

3. Графові задачі

Задачі, пов’язані з графами, є важливою частиною олімпіадних змагань. Графи використовують для моделювання складних структур, таких як мережі, маршрути, зв’язки між об’єктами тощо. У таких задачах учасники повинні знайти певні властивості графів, наприклад, визначити наявність циклів, знайти найкоротші шляхи або максимальний потік. Задачі на графи зазвичай включають використання класичних алгоритмів, таких як алгоритм Дейкстри, алгоритм пошуку в глибину (DFS) та в ширину (BFS), алгоритми для побудови мінімального остовного дерева.

Типові методи розв’язання включають:

— пошук у глибину та в ширину для аналізу зв’язності графів;
— алгоритм Дейкстри для пошуку найкоротших шляхів;
— алгоритм Флойда — Воршелла для обчислення всіх пар найкоротших шляхів;
— алгоритми для знаходження компонент зв’язності та топологічного сортування.

4. Динамічне програмування

Задачі на динамічне програмування є одними з найскладніших і часто зустрічаються на олімпіадах з інформатики. Вони зазвичай вимагають вирішення задач, які можна розбити на підзадачі з перекриттям, і оптимізацію рішення через збереження результатів цих підзадач для уникнення повторних обчислень. Типові приклади задач на динамічне програмування включають: задачу про рюкзак, задачі на пошук найдовшої загальної послідовності та інші.

Типові методи розв’язання включають:

— мемоізація (збереження результатів підзадач для їх повторного використання);
— рекурсивне розбиття задачі на менші підзадачі з використанням збереження результатів;
— покрокова побудова таблиці для обчислення оптимальних рішень.

5. Пошук і сортування

Задачі на пошук і сортування — це класика алгоритмічних змагань, і вони часто зустрічаються в першій частині олімпіад. Вони включають в себе не лише класичні алгоритми сортування, такі як швидке сортування, сортування злиттям чи сортування бульбашкою, але й пошукові алгоритми, такі як бінарний пошук, пошук у відсортованому масиві або пошук підмасиву в рядку. Метою таких задач є не лише реалізація алгоритму, але й оптимізація роботи з великими даними та часова складність.

Типові методи розв’язання включають:

— алгоритми сортування (наприклад, бульбашкове сортування, швидке сортування, сортування злиттям);
— бінарний пошук для знаходження елементів у відсортованих масивах;
— пошук підмасивів у рядках за допомогою алгоритмів на основі хешування або рівнянь.

Кожен тип задач потребує специфічного підходу до розв’язання, що дозволяє учням розвивати логічне мислення, аналітичні здібності та навички програмування, важливі для успішної участі у змаганнях та для розвитку кар’єри в ІТ-сфері [17].

Варто звернути увагу на два основні підходи до їх розв’язання, що визначають ефективність вирішення задач: підхід «у лоб» і оптимізовані алгоритми. Ці підходи не тільки демонструють різницю в рівні складності розв’язку, а й вимагають від учнів різних навичок і стратегій. Вибір методу залежить від характеру задачі, доступного часу та необхідної точності результату.

Підхід «у лоб» полягає в прямому розв’язанні задачі без особливих спроб оптимізації або застосування складних алгоритмів. Зазвичай, це означає вирішення проблеми шляхом перебору всіх можливих варіантів, послідовного виконання дій, не зважаючи на можливі повторення чи зайві операції. Це може бути ефективним для задач малого розміру або у випадках, коли обсяг вхідних даних обмежений, що дозволяє швидко отримати відповідь. Однак для великих обсягів даних або задач з високими вимогами до часу виконання та пам’яті цей підхід може бути дуже неефективним і призвести до перевищення ліміту часу або пам’яті, встановленого на олімпіаді.

Для прикладу, розв’язання задачі на пошук найменшого елемента в масиві або перевірку, чи є число простим, може бути здійснено шляхом простого перебору всіх елементів чи перевірки кожного можливого дільника. У таких випадках відсутність оптимізації може не викликати проблем. Проте, коли задача стає складнішою, наприклад, при пошуку в масиві величезного розміру, або при обробці чисел великої довжини, такий підхід стає неефективним [16].

З іншого боку, оптимізовані алгоритми — це підхід, який включає використання спеціальних математичних та програмних методів для покращення ефективності розв’язку задачі. Це може бути досягнуто через зменшення кількості операцій, пам’яті, що використовується, чи часу, необхідного для виконання програми. Оптимізація може включати використання більш складних алгоритмів, таких як бінарний пошук замість лінійного, застосування методів динамічного програмування, що дозволяють зберігати результати попередніх обчислень, або використання хеш-таблиць для швидкого доступу до даних.

Для задач, де час і пам’ять обмежені, оптимізація стає необхідною умовою для успішного розв’язку. Наприклад, задача на знаходження найбільшого спільного підрядка в двох рядках може вимагати використання алгоритму з динамічним програмуванням, що значно скорочує кількість

необхідних операцій у порівнянні з підходом «у лоб», який може потребувати перебору всіх можливих варіантів підрядків.

Звичайно, що перехід до складніших задач вимагає від учасників не лише глибоких теоретичних знань, а й уміння застосовувати сучасні методи та стратегії для їх ефективного розв'язання. Якщо на попередніх етапах олімпіадних змагань учасники можуть обмежуватися стандартними алгоритмами та підходами, то в складніших задачах необхідно використовувати більш просунуті методи, що дозволяють досягти оптимальних результатів навіть за складних обмежень по часу і пам'яті. Сучасні методи та стратегії розв'язання таких задач часто базуються на комбінації кількох підходів, включаючи теорію графів, комбінаторні методи, динамічне програмування та інші інноваційні техніки [16].

Одним із основних сучасних методів є розділення задачі на підзадачі. Цей підхід використовує принцип «розділяй і володарюй», який дозволяє значно спростити складну задачу, розбиваючи її на менші частини, що є легшими для розв'язання. Для прикладу, у задачах на пошук шляху або оптимізацію може бути корисним застосування алгоритмів пошуку в графах або пошуку з обмеженнями, які здатні обробити великий обсяг даних за допомогою розподілу задачі на менш складні етапи.

Ще однією важливою стратегією є використання динамічного програмування, яке дозволяє ефективно вирішувати задачі з перекриттям підзадач. У таких випадках можна уникнути надмірних обчислень, зберігаючи вже отримані результати і використовуючи їх для розв'язання складніших частин задачі. Це особливо корисно для задач, що вимагають оптимізації, наприклад, пошук найкоротшого шляху або максимального потоку в графах, а також для комбінаторних задач, де на кожному кроці з'являється велика кількість можливих варіантів.

Додатково, для складних задач, що вимагають багатокрокового пошуку або обробки великих обсягів даних, застосовують алгоритми з жадібними методами (greedy algorithms). Вони передбачають на кожному кроці вибір

найкращого локального рішення, яке забезпечує наближення до глобально оптимального. Такі алгоритми широко використовують у задачах на побудову дерев, графів і при пошуку мінімальних остовних дерев [15].

Методи пошуку в глибину та ширину теж залишаються важливими для вирішення складних задач на графах, особливо коли необхідно визначити наявність зв’язку між елементами, розрахувати компоненти зв’язності, або здійснити пошук шляхів в умовах обмежень.

Не менш важливим є метод евристичних алгоритмів (наприклад, генетичні алгоритми, алгоритм мурашиного колонії), які застосовують для пошуку наближених рішень у випадках, коли оптимальне рішення важко знайти за допомогою традиційних методів через складність задачі. Ці методи зазвичай застосовують для задач, що мають великі обсяги можливих варіантів, де перебір усіх варіантів є непрактичним [24].

У контексті сучасних змагань і олімпіад велике значення має також уміння застосовувати паралельні обчислення та розподілені алгоритми, особливо коли задача потребує обробки величезних обсягів даних або швидкої реакції на численні вхідні дані. Ці стратегії відкривають нові можливості для швидкого розв’язання складних задач в умовах обмежених ресурсів.

Загалом, сучасні методи та стратегії розв’язання складних олімпіадних задач відрізняються значною різноманітністю і гнучкістю, дозволяючи учасникам вибирати оптимальні підходи залежно від умов задачі. Використання передових алгоритмів і стратегій у поєднанні з математичними методами забезпечує успіх у вирішенні навіть найскладніших проблем.

1.4. Поява і розвиток напрямку «Інформаційні технології»

У контексті розвитку олімпіадного руху з інформатики за останнє десятиліття відзначилося появою нового напряму — «Інформаційні технології», який, хоча й виник у тісному зв’язку з класичною інформатикою, має низку істотних відмінностей. Перехід від виключно алгоритмічного мислення до практичного застосування цифрових інструментів став відповіддю на актуальні

запити часу: стрімке поширення цифрових сервісів, програмного забезпечення для користувача рівня, хмарних рішень і автоматизації повсякденних процесів.

Класична інформатика в олімпіадному форматі зосереджена переважно на програмуванні та алгоритмах. Її головною метою є виявлення рівня володіння учнями структурованим мисленням, вмінням будувати та аналізувати алгоритми, вирішувати задачі з високим рівнем абстракції, що часто базуються на математичному апараті. Завдання з класичної інформатики потребують глибоких знань мов програмування (C++, Python, Pascal тощо), розуміння структур даних, теорії графів, комбінаторики, динамічного програмування, логіки й інших теоретичних основ.

Натомість напрям «Інформаційні технології» фокусується на застосуванні ІТ-засобів у практичних ситуаціях. Цей напрям більше тяжіє до інженерного мислення і менш до абстрактного аналізу. Основними складовими завдань у цьому напрямку є робота з текстовими процесорами, електронними таблицями, базами даних, презентаціями, створення простих вебсторінок, використання графічних редакторів та інтеграція різноманітних сервісів. Такий формат підготовки та змагань покликаний відобразити реальні умови, у яких випускники працюватимуть у сучасному цифровому середовищі [6].

Ще одна суттєва відмінність полягає у критеріях оцінювання. У класичній інформатиці головним критерієм є коректність та ефективність розв’язку задачі, відповідність алгоритму часовим та пам’яттєвим обмеженням. В олімпіадах з ІТ важливими стають точність виконання технічного завдання, функціональність та естетичність створеного продукту, а також дотримання певних стандартів оформлення.

Доцільно детальніше зупинитися на тих інструментах і технологіях, що формують зміст завдань в ІТ-олімпіадах. Вони відображають практичну спрямованість цього напряму, орієнтовану на цифрову грамотність, автоматизацію типових завдань, презентацію інформації та роботу з даними. Центральну роль у цьому контексті відіграють офісні пакети програм, бази

даних, інструменти для створення презентацій, а також базові засоби веброзробки.

Офісні програми, зокрема текстові редактори (Microsoft Word, Google Docs, LibreOffice Writer тощо), традиційно є першою сходинкою у підготовці до ІТ-олімпіад. Від учасників вимагається не лише вміння вводити текст, а й форматувати його згідно з певними вимогами: оформлення заголовків, нумерація, створення змісту, вставлення графіків, таблиць, зображень, посилань, колонтитулів, а також використання стилів і шаблонів. Важливою є увага до типографіки, однаковості оформлення та здатності виконувати дії ефективно, з використанням автоматизованих функцій (наприклад, автозаповнення змісту, посилань, списків, формулювання полів).

Другим важливим компонентом є електронні таблиці (Microsoft Excel, Google Sheets тощо). Завдання в цьому підрозділі охоплюють широкий спектр дій: від базового введення даних та форматування до використання формул, логічних функцій, умовного форматування, побудови діаграм, фільтрації, сортування та створення зведених таблиць. В більш складних випадках застосовуються інструменти автоматизації, макроси або елементи скриптів, що дозволяють моделювати дані або вирішувати практичні задачі — наприклад, бухгалтерські, статистичні чи управлінські [4].

Особливе місце займає робота з базами даних, яка набуває дедалі більшої актуальності в олімпіадному русі. Учасникам необхідно продемонструвати розуміння структури бази даних, нормалізації, створення таблиць, зв’язків між ними, а також уміти формувати запити за допомогою мови SQL. Такі завдання включають створення запитів для вибірки, обчислень, об’єднання таблиць, сортування та групування. Вони не лише перевіряють знання синтаксису, а й навички логічного мислення та здатність працювати з великою кількістю взаємопов’язаних даних. Нерідко оцінюється також уміння реалізувати інтерфейс для доступу до бази даних або перенесення її до іншого середовища (наприклад, у вебпроєкт).

Інструменти для створення презентацій (Microsoft PowerPoint, Google Slides тощо) теж є важливою частиною ІТ-змагань. Завдання в цьому блоці спрямовані на перевірку умінь структурувати інформацію, використовувати візуальні ефекти (анімації, переходи), дотримуватися дизайну слайдів, логіки подання інформації, інтерактивності. Додатково можуть перевірятися вміння роботи з графікою, діаграмами, мультимедійними елементами, посиланнями, управління шаблонами та стилями. Важливим є також узгодження оформлення з тематикою доповіді — поєднання технічних навичок з презентаційною культурою.

Ще одним компонентом сучасного ІТ-напряму є початкові навички у вебтехнологіях. Учнів навчають створювати прості вебсторінки за допомогою HTML та CSS, іноді з елементами JavaScript. Завдання включають верстку сторінки за заданим макетом, стилізацію елементів, додавання зображень, таблиць, гіперпосилань, форм тощо. Цей блок підсилює міждисциплінарний підхід у підготовці, адже потребує базових знань з комп’ютерної графіки, логіки й дизайну користувацьких інтерфейсів.

Таким чином, роль офісних програм, баз даних, презентаційних і вебінструментів у напрямі «Інформаційні технології» не є другорядною. Вони відображають зміщення акценту в олімпіадному русі до практичних умінь, що можуть бути безпосередньо застосовані в повсякденній роботі, навчанні, а згодом — і в професійній діяльності. Розвиток саме цих навичок сприяє формуванню цифрової компетентності учнів та забезпечує широкий спектр можливостей для подальшої спеціалізації.

1.5. Формування ключових навичок, необхідних для успішної участі в олімпіадах

Формування ключових навичок, необхідних для успішної участі в олімпіадах з інформатики, є результатом тривалого, системного та багатоаспектного навчального процесу. Як вже зазначалося, сучасні олімпіади, особливо з класичної інформатики, ставлять перед учнями складні задачі, які

вимагають не лише технічного володіння мовою програмування, а й розвинутого логічного та алгоритмічного мислення. Ці когнітивні й технічні компоненти тісно переплетені між собою і визначають успішність учасника незалежно від формату змагання — шкільного, міського, обласного, всеукраїнського чи міжнародного рівня [18].

Логічне мислення є базисом всієї інформатичної підготовки. Саме воно дозволяє учню аналізувати умову задачі, виділяти ключові елементи, встановлювати зв’язки між даними, виявляти приховані закономірності та суперечності. Завдяки логічному мисленню формуються первинні навички аналізу вхідних даних і прогнозування результатів. Воно є передумовою для побудови коректного алгоритму, адже допомагає структурувати розв’язок, розділити завдання на підзадачі, виокремити базові випадки й передбачити винятки. Логіка особливо важлива у задачах на істинність, побудову висловлювань, перевірку правильності розв’язку та формування перевірочних тестів.

Алгоритмічне мислення тісно пов’язане з логічним, однак має свою специфіку. Воно полягає в умінні послідовно формалізувати розв’язання задачі у вигляді чіткої, не суперечливої послідовності дій, яку згодом можна реалізувати засобами програмування. Розвиток алгоритмічного мислення передбачає знайомство з базовими алгоритмами (лінійними, розгалуженнями, циклами), поступове освоєння алгоритмів обробки масивів, рядків, графів, рекурсивних структур, а також здатність узагальнювати й модифікувати відомі алгоритми під нові умови. Це мислення також включає здатність до оцінювання ефективності алгоритму — зокрема його часової і просторової складності [21].

Навички програмування — це практичне втілення логічного та алгоритмічного мислення в реальних умовах. Без добре сформованої технічної бази навіть найкращий алгоритм залишиться лише теоретичною конструкцією. Тому підготовка до олімпіад передбачає ґрунтовне вивчення мови програмування (найчастіше — C++, Python або Java), включаючи її синтаксис, роботу з типами даних, бібліотеками, особливостями обробки помилок,

компіляції, оптимізації тощо. Важливо, щоб учень умів писати чистий, читабельний код, користувався коментарями, розбивав програму на функціональні блоки, умів тестувати власний код і знаходити в ньому помилки (debugging).

Таблиця 1.1

Ключові навички для участі в олімпіадах з інформатики

Навичка	Змістовне наповнення	Очікувані результати
Логічне мислення	Аналіз умови задачі, побудова логічних ланцюжків, виявлення суперечностей	Здатність до структурного аналізу задачі, пошук закономірностей, логічна перевірка рішень
Алгоритмічне мислення	Побудова алгоритмів, формалізація розв’язків, аналіз ефективності алгоритмів	Уміння вибудовувати послідовні дії, оптимізувати розв’язки, використовувати відомі алгоритми
Програмування	Володіння мовою, написання коду, робота з середовищем, відлагодження	Практична реалізація розв’язку, тестування програм, робота з пам’яттю, оптимізація продуктивності

Комплексне формування цих навичок можливе тільки за умов системного підходу до навчання, що передбачає поступове ускладнення задач, роботу з помилками, аналіз прикладів та регулярну практику. Особливо важливою є інтеграція всіх трьох навичок в одне ціле: лише тоді учень здатен не тільки розв’язувати задачі, а й робити це впевнено, ефективно й оптимально — тобто на рівні, що відповідає вимогам сучасних олімпіад.

Наступним критично важливим етапом підготовки до олімпіад є розвиток навичок самостійної роботи. Успішна участь в олімпіадах з інформатики майже неможлива без вміння вчитися самостійно, аналізувати свої помилки,

опрацьовувати нові теми поза межами шкільної програми та брати участь у регулярних тренуваннях, які моделюють реальні умови змагання.

Самостійне навчання охоплює як опрацювання теоретичного матеріалу з нових тем, так і дослідження прикладів розв’язання задач. У цьому контексті дуже важливу роль відіграє здатність працювати з документацією мов програмування, інтернет-ресурсами (як-от GeeksforGeeks, Stack Overflow, Coursera, Stepik), офіційними архівами задач олімпіад, методичними посібниками та підручниками. Учень повинен навчитися самостійно шукати потрібну інформацію, перевіряти її достовірність, тестувати отримані знання на практиці. Такий підхід формує вміння планувати власний навчальний маршрут, визначати слабкі сторони та компенсувати їх за допомогою доступних інструментів.

Розбір задач є не лише засобом перевірки знань, а й одним з найефективніших способів навчання. Учень, який аналізує задачу, повинен пройти шлях від уважного читання умови — через побудову алгоритму — до реалізації та тестування коду. Успішна підготовка передбачає не просто механічне відтворення відомих рішень, а й глибоке розуміння логіки задачі, оцінку альтернативних підходів, а також відпрацювання найбільш ефективного з них. Важливим є і розбір чужих розв’язків — з відкритих архівів олімпіад або тематичних форумів — де можна побачити різні стилі кодування, підходи до оптимізації, оформлення програм. Розбір задач має включати в себе аналіз помилок, порівняння варіантів рішень за часом виконання, використанням пам’яті, простотою реалізації [15].

Участь у тренуваннях — завершальна складова цього процесу — дозволяє перевірити сформовані знання в умовах, максимально наближених до реальних олімпіад. Індивідуальні, групові або командні тренування на онлайн-платформах (таких як E-olymp, Codeforces, AtCoder, LeetCode, CSES) не лише розвивають технічні навички, а й навчають працювати в обмежений час, швидко аналізувати умову, правильно обирати порядок розв’язування задач. Участь у віртуальних змаганнях також тренує психологічну стійкість, здатність

не втрачати концентрацію після невдалого рішення або складної задачі. Систематичні тренування, що включають підсумковий аналіз рішень, корекцію стратегії й роботу над помилками, значно підвищують загальний рівень готовності учня [9].

В цілому можна зробити висновок: самостійне навчання, систематичний розбір задач і регулярна участь у тренуваннях створюють основу для індивідуального прогресу учня. Проте попри їхню важливість, ефективна підготовка до олімпіад неможлива без організованої підтримки вчителя та навчального середовища, що виходить за межі стандартного уроку інформатики. Саме позаурочна діяльність — у форматі гуртків, факультативів, спецкурсів і навчально-тренувальних зборів — відіграє визначальну роль у формуванні високого рівня готовності учнів до олімпіад.

Під час уроків інформатики, навіть у класах з поглибленим вивченням предмета, вкрай обмежені можливості для розгляду складних олімпіадних задач або застосування нестандартних підходів до програмування. Типова програма зосереджена на базових знаннях, адаптованих до рівня середнього учня, і не передбачає інтенсивної практики в стилі олімпіад. Натомість саме гуртки з програмування дозволяють поглибити знання в індивідуальному темпі, адаптуючи навчання до потреб здібних учнів, які прагнуть більшого [9].

У рамках гурткової роботи значну увагу приділяють практичному розв’язанню задач із різних тематичних блоків — наприклад, графові алгоритми, динамічне програмування, задачі на пошук та оптимізацію, комбінаторні техніки тощо. Такі заняття зазвичай проходять у менш формальній атмосфері, що сприяє відкритій комунікації між учнями і викладачем, спільному обговоренню рішень, формуванню командної взаємодії. Більшість сильних олімпіадників, згідно з дослідженнями методистів і практикою викладачів, саме в рамках гурткової діяльності вперше стикаються з олімпіадними задачами підвищеного рівня складності.

Окрему цінність становлять спеціалізовані курси та навчальні збори — як шкільного, так і регіонального чи національного рівня. На таких заходах, що

часто тривають кілька днів або тижнів, учні мають змогу інтенсивно працювати з досвідченими тренерами, слухати лекції з алгоритмів, брати участь у змаганнях і аналізувати типові помилки. Такі форми навчання моделюють реальний досвід підготовки до міжнародних олімпіад, де без тривалої позашкільної роботи успіх практично недосяжний.

Загалом, систематична робота поза межами уроків є незамінною умовою ефективної підготовки до олімпіад з інформатики. Саме вона дозволяє реалізувати індивідуальний навчальний маршрут, створити умови для глибокого занурення в предмет, забезпечити доступ до кваліфікованих менторів і сформувати стійку мотивацію до вдосконалення. Поєднання внутрішньої самостійності учня з підтримкою позашкільних освітніх форм дає змогу досягти високих результатів у конкурентному олімпіадному середовищі.

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

У результаті аналізу сучасного стану та тенденцій розвитку олімпіадного руху з інформатики з’ясовано, що цей напрямок має не лише освітнє, але й стратегічне значення для формування майбутніх ІТ-фахівців. Встановлено, що зміст конкурсних завдань поступово ускладнюється, відображаючи динаміку розвитку інформаційних технологій та потребу в нових компетентностях. Погляди науковців і методистів підтверджують необхідність системної підготовки учнів, де поєднуються класичні підходи до формування алгоритмічного мислення з сучасними трендами, пов’язаними із застосуванням ІТ. Окрему увагу приділено аналізу типів олімпіадних завдань і методів їхнього розв’язання, що дозволяє вибудувати чітку структуру підготовки. Узагальнення матеріалів розділу дає змогу перейти від теоретичного обґрунтування проблеми до практичного пошуку ефективних організаційних форм навчання.

РОЗДІЛ 2. МЕТОДИКА І ПРАКТИКА ПІДГОТОВКИ УЧНІВ ДО ОЛІМПІАД З ІНФОРМАТИКИ

2.1. Організаційні форми підготовки: уроки, гуртки, спецкурси

В практичному аспекті підготовки учнів до олімпіад з інформатики ключову роль відіграє ефективне планування і використання навчального часу. Уроки інформатики як базова форма навчання в шкільній системі мають потенціал стати стартовим майданчиком для розвитку олімпіадних навичок. Проте слід чітко розуміти як можливості, так і обмеження, притаманні цій формі організації навчального процесу.

По-перше, шкільні уроки інформатики за типовою програмою в основному спрямовані на формування базових знань і навичок роботи з комп’ютером, засобами автоматизації офісної діяльності, основами алгоритмізації, структурою програмування (залежно від рівня). У класах, де інформатика вивчається на стандартному рівні (1 година на тиждень у базовій школі, 1–2 години — у старшій), часу для глибокої олімпіадної підготовки фактично недостатньо. Урок має жорстку структуру, обмежену в часі (45 хвилин), часто передбачає фронтальне викладання з невеликим обсягом самостійної практики, а головне — орієнтований на середній рівень учнів [17].

Однак навіть в умовах обмеженого часу вчитель, який має відповідну підготовку та педагогічну гнучкість, може поступово закладати в уроки основи олімпіадної тематики. Для прикладу, у темі «Алгоритми сортування» вчитель може дати завдання не лише на реалізацію простого сортування (bubble sort), а й запропонувати задачу з підвищеною складністю, де треба визначити, який із кількох алгоритмів спрацює швидше на великих даних. Водночас урок може містити невеликий аналіз завдання з реально проведеної олімпіади (наприклад, переформульоване для рівня учня), що пробуджує інтерес і демонструє приклад використання знань на практиці [9].

У 9–11 класах, де програмування вже є частиною навчальної програми, з’являється більше можливостей для введення олімпіадних задач у зміст уроку. Наприклад, при вивченні рекурсії, динамічних структур даних або алгоритмів

пошуку вчитель може організувати міні-змагання, де учні в групах шукають найоптимальніше рішення для задачі. Така форма роботи формує змагальний дух і аналітичне мислення — два ключових елементи олімпіадної мотивації.

Важливо враховувати, що навіть якщо програма не передбачає спеціальних олімпіадних тем, окремі хвилини на уроці можна присвятити поясненню прийомів оптимізації коду, розбору «пасток» у задачах, демонстрації реальних прикладів змагань, або хоча б знайомству з платформами типу CSES, E-olymp, Codeforces. Учнів можна залучити до реєстрації та проходження тренувальних задач як домашнього завдання або додаткової діяльності.

Уроки, також, можуть бути корисними для виявлення здібних учнів, яким цікаво розв’язувати складніші завдання. Саме під час таких занять учитель має змогу спостерігати за тим, хто швидко орієнтується у новому матеріалі, шукає нестандартні підходи або проявляє зацікавлення задачами, які виходять за межі стандарту. Такі учні можуть стати ядром майбутньої олімпіадної групи, яку варто запрошувати до участі в гуртках, факультативах або проектній діяльності.

Також, не варто нехтувати використанням міжурочного простору: самостійні роботи, творчі домашні завдання, додаткові консультації після уроків. Для прикладу, після теми «Умовні оператори» учням можна дати задачу на аналіз діапазонів чисел з кількома гілками логіки, що за складністю ближче до початкового рівня олімпіад [5].

У підсумку можна відмітити, що урок інформатики, хоча й має обмежені ресурси, може стати стартовим майданчиком для впровадження олімпіадної тематики — через адаптацію задач, посилення мотиваційної складової, спостереження за здібностями учнів і подальше залучення їх до позаурочної діяльності. Ключовим чинником при цьому є компетентність учителя, його методична свобода та стратегічне бачення підготовки обдарованих учнів.

У тих випадках, коли стандартний навчальний час на уроках інформатики вичерпується, а інтерес учня до предмета лише зростає, найприроднішим і найефективнішим способом підтримати та розвинути цей інтерес є залучення

до позакласної діяльності. Саме факультативи, гуртки й спецкласи створюють умови для глибшого занурення у світ олімпіадного програмування, де вже не діють суворі рамки шкільної програми, а кількість часу, уваги до індивідуальних запитів та рівень завдань значно ширші.

Факультативні курси, які часто вводять у старших класах, дозволяють викладачеві формувати навчальний зміст навколо олімпіадної тематики. Теми можуть охоплювати розширену алгоритміку, складні методи оптимізації, нестандартні задачі з комбінаторики, пошуку, теорії чисел, динамічного програмування. Факультатив — це місце, де учні мають змогу не просто прослухати теоретичний матеріал, а розв’язати десятки задач, обговорити різні підходи, провести тестування своїх програм, і що найголовніше — помилятися і вчитися на власних помилках без страху «поганої оцінки» [15].

Гурткова форма роботи більш гнучка за своєю структурою і дозволяє організовувати заняття не лише для учнів одного класу, а й для учнів різного віку і рівня підготовки. В таких умовах легко створити «вертикальну» модель навчання: досвідчені учасники можуть допомагати новачкам, виконуючи функцію асистентів, тьюторів або навіть тренерів. Це формує культуру командної роботи, взаємопідтримки й послідовної передачі знань. До того ж, саме гуртки найчастіше стають стартовим майданчиком для участі в перших шкільних, районних і обласних олімпіадах [9].

Підготовка в гуртках має низку практичних переваг. По-перше, у такій формі можна застосовувати регулярні тренування у форматі «контрольного змагання», коли учням пропонують набір із 3–5 задач на час, а після завершення — проводиться спільний розбір, порівняння рішень, обговорення алгоритмів. Це створює атмосферу, максимально наближену до справжньої олімпіади, дозволяє учням звикати до психологічного навантаження, вчитись розподіляти час, переключатись між задачами, уникати технічних помилок.

Крім того, у гуртках можна використовувати елементи гейміфікації та довготривалих челленджів — наприклад, «турніри» між учнями, «щотижневі рейтинги розв’язаних задач» або завдання з поступовим ускладненням (task

ladder). Це мотивує школярів тримати темп, порівнювати власні результати з однолітками і прагнути покращення.

Особливо ефективною є діяльність спеціалізованих класів або профільних груп, які формуються при ліцеях, гімназіях, іноді навіть на базі вищих навчальних закладів. У таких класах інформатика викладається кілька разів на тиждень, нерідко доповнюється додатковими годинами на практику, а програма значно виходить за межі типового шкільного курсу. Тут викладачі мають змогу реалізовувати складні навчальні модулі, наприклад, «Застосування жадібних алгоритмів», «Теорія графів і пошук шляхів», «Аналіз складності алгоритмів», тощо з поєднанням коротких лекцій, великої кількості задач і постійного самостійного кодування.

Позакласна діяльність, на відміну від обов’язкових уроків, створює умови для формування індивідуального маршруту навчання. Тут учень може просуватись у зручному темпі, працювати з матеріалом, що відповідає саме його рівню, і фокусувати увагу на тих темах, які викликають інтерес або складність. До того ж, на гуртках і спецкурсах зазвичай використовуються сучасні онлайн-платформи для змагань і тренувань, такі як E-olymp, Codeforces, CSES, AtCoder, які дозволяють вести реальну статистику, стежити за прогресом і готуватись до офіційних етапів олімпіад на справжніх змагальних системах [8].

Одним з найбільш дієвих результатів правильно організованої позакласної роботи — у форматі гуртків, факультативів чи спецкурсів — є можливість впровадження системного і довготривалого плану підготовки до олімпіади. Якщо заняття проводяться випадковим чином, без чіткого уявлення про послідовність вивчення тем і розвиток навичок, ефективність такої роботи буде значно нижчою. Саме тому на практиці варто орієнтуватися на модельний план підготовки, який дозволяє поетапно формувати знання, поступово нарощувати складність задач, розвивати стійкі алгоритмічні навички та забезпечити контроль прогресу.

Модельний план підготовки — це структурований річний або піврічний графік занять, складений відповідно до вікових можливостей учнів, рівня їхньої

підготовки та етапів олімпіадного циклу. Він передбачає як тематичне, так і рівневе планування: що вивчається (які алгоритмічні теми) і якого рівня задачі використовуються (базові, середні чи олімпіадні високого рівня). Зазвичай такий план складається з 30–35 занять тривалістю по 90–120 хвилин (у форматі гуртка або факультативу), розділених на блоки [22].

Перший блок — вступно-базовий, охоплює повторення основних понять: введення-виведення, робота з масивами, цикли, умовні оператори, основи рекурсії. Завдання на цьому етапі мають бути простими, але з чіткою логічною структурою, щоб учні звикали до формату задач. Тут важливо закласти навички «читання умови», розбору прикладів, побудови найпростішої моделі розв’язання. Особлива увага приділяється акуратності оформлення коду, тестуванню та аналізу.

Другий блок — системне вивчення алгоритмів, що охоплює найбільш уживані типи задач: пошук у масиві, бінарний пошук, сортування, префіксні суми, жадібні алгоритми, двовимірні масиви, основи теорії чисел. Для кожної теми рекомендується проводити не лише пояснення, а й щонайменше 2 тренувальні заняття з розв’язуванням задач від простих до середнього рівня. На цьому етапі учні набувають досвіду переходу від «лобових» рішень до ефективніших.

Третій блок — поглиблений алгоритмічний рівень, на якому вивчаються складніші методи: рекурсія із запам’ятовуванням (динамічне програмування), алгоритми на графах (обхід у глибину, у ширину, пошук компонент), відсортовані структури даних (множини, карти, пріоритетні черги), бітові маски, обробка рядків, техніка «двох вказівників», техніка «розбиття на блоки». Заняття у цьому блоці можуть бути побудовані навколо ключових задач-еталонів: спочатку розбір алгоритму, потім кілька задач на аналогічну ідею, поступово зростаючої складності.

Четвертий блок — тренувальний, коли основну частину часу займають змагання у форматі «контесту» (3–4 задачі на 2 години), а після — обов’язковий колективний розбір, під час якого учні вчаться шукати

оптимальні рішення, порівнювати ефективність різних підходів, знаходити помилки. Тут корисно також використовувати задачі з попередніх олімпіад різних рівнів (район, область, всеукраїнська, міжнародні онлайн-турніри), адаптуючи їх до рівня групи.

Окремий акцент у модельному плані робиться на роботу з онлайн-платформами, такими як CSES, E-olymp, Timus, Codeforces. План має включати щотижневі домашні завдання або онлайн-контести, які учні виконують індивідуально. Результати обговорюються на наступному занятті. Це створює безперервний зв'язок між гуртком і самостійною діяльністю, а також тренує навички, необхідні для реальних змагань [4].

Наприкінці навчального циклу обов'язковим є модуль підготовки до олімпіади — це симуляція змагань, перевірка роботи з тестами, контроль часу, психологічна готовність до виступу. Можна організувати кілька пробних турів, у тому числі з зовнішнім суддівством, щоб учні адаптувались до олімпіадної атмосфери.

Отже, модельний план підготовки — це не просто перелік тем, а інструмент педагогічного управління процесом навчання, що поєднує алгоритмічну логіку, психолого-педагогічну послідовність і практичну ефективність. Він дозволяє поступово формувати стійкі навички, відслідковувати індивідуальний прогрес кожного учня, створювати умови для зростання сильних учасників і підвищувати загальний рівень підготовки шкільної команди.

2.2. Застосування онлайн-платформ для тренування

Онлайн-платформи для тренування з олімпіадного програмування вже давно стали не просто додатковим інструментом, а незамінним компонентом ефективної підготовки. У сучасному навчальному середовищі, де темп роботи високий, а рівень конкуренції на олімпіадах постійно зростає, використання онлайн-ресурсів дозволяє значно розширити часові й тематичні межі занять. Учень отримує можливість практикуватися у будь-який зручний час, з необмеженим доступом до задач, тестів і систем автоматичного оцінювання.

У таблиці 2.1 подано докладну характеристику п’яти найпопулярніших платформ, які активно використовуються у практиці підготовки до олімпіад, як в Україні, так і в міжнародному середовищі.

Таблиця 2.1

Онлайн-платформи для тренування з олімпіадного програмування

Назва платформи	Особливості	Рівень задач	Мова інтерфейсу	Підтримка мов програмування
E-olymp	Український онлайн-архів задач, включає архіви олімпіад, має систему тестування	Від початкового до складного	Українська, англійська	C++, Python, Java, Pascal тощо
Codeforces	Платформа з регулярними контестами, система рейтингів, активне міжнародне ком’юніті	Від середнього до дуже високого	Англійська, підтримка кирилиці	C++, Python, Java, Kotlin та ін.
Codeforces	Платформа з регулярними контестами, система рейтингів, активне міжнародне ком’юніті	Від середнього до дуже високого	Англійська, підтримка кирилиці	C++, Python, Java, Kotlin та ін.
Timus Online Judge	Класична платформа зі стабільним набором задач	Від середнього до високого	Англійська	C++, Java, Pascal, Python
CSES Problem Set	Систематизований набір задач за темами, з хорошими	Від початкового до	Англійська	C++, Python, Java, Rust

	поясненнями	середнього		

Кожна з платформ має свою унікальну специфіку, і тому їхнє використання варто адаптувати до етапу підготовки. Для прикладу, E-olymp — це ідеальний старт для новачків. Інтерфейс повністю зрозумілий українському школяреві, є архіви попередніх олімпіад (у тому числі районного та обласного рівня), а також задачі з українськими умовами. Завдяки цьому учні можуть практикуватися в максимально наближеному до реальних змагань контексті, використовуючи знайомі формати введення/виведення [10].

На більш просунутому рівні рекомендується використовувати Codeforces — платформу з активною міжнародною спільнотою, великою кількістю тематичних задач, рейтингами та щотижневими контестами. Її перевага — у наявності величезного архіву задач різної складності, які вже розв'язувались багатьма учасниками, тому існує змога не лише перевірити своє рішення, а й проаналізувати чужі ідеї. Крім того, на форумі можна знайти тематичні добірки задач за алгоритмами, що дозволяє проводити цілеспрямовану підготовку [11].

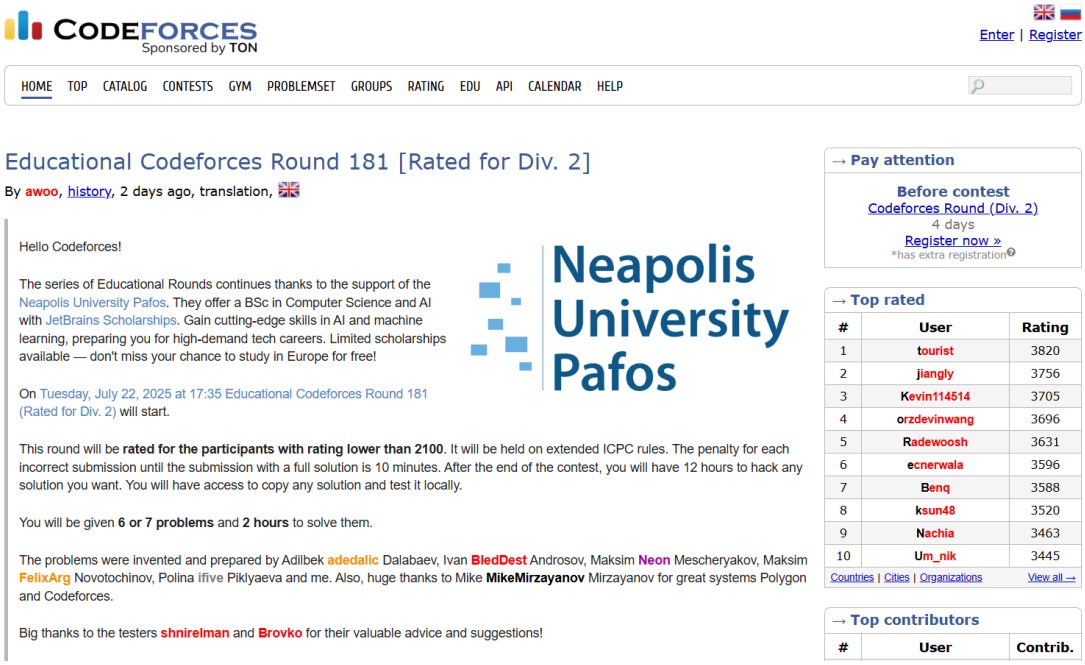

Рис. 2.1 Інтерфейс платформи Codeforces

Timus — платформа, що зберігає стабільну репутацію ще з початку 2000-х років. Її головна цінність — у класичних задачах без зайвих відволікаючих елементів. Це ідеальне середовище для тренування «на швидкість»: умови короткі, інтерфейс мінімалістичний, можна легко зосередитися на суті задачі.

CSES — особливо корисна для вивчення алгоритмів тематично. Наприклад, блок «Sorting and Searching» містить задачі в чіткій прогресії складності — від базових до складних. Завдяки цьому платформа добре підходить для індивідуального домашнього навчання, особливо якщо мета — послідовно опрацювати конкретну тему (наприклад, динамічне програмування чи графи).

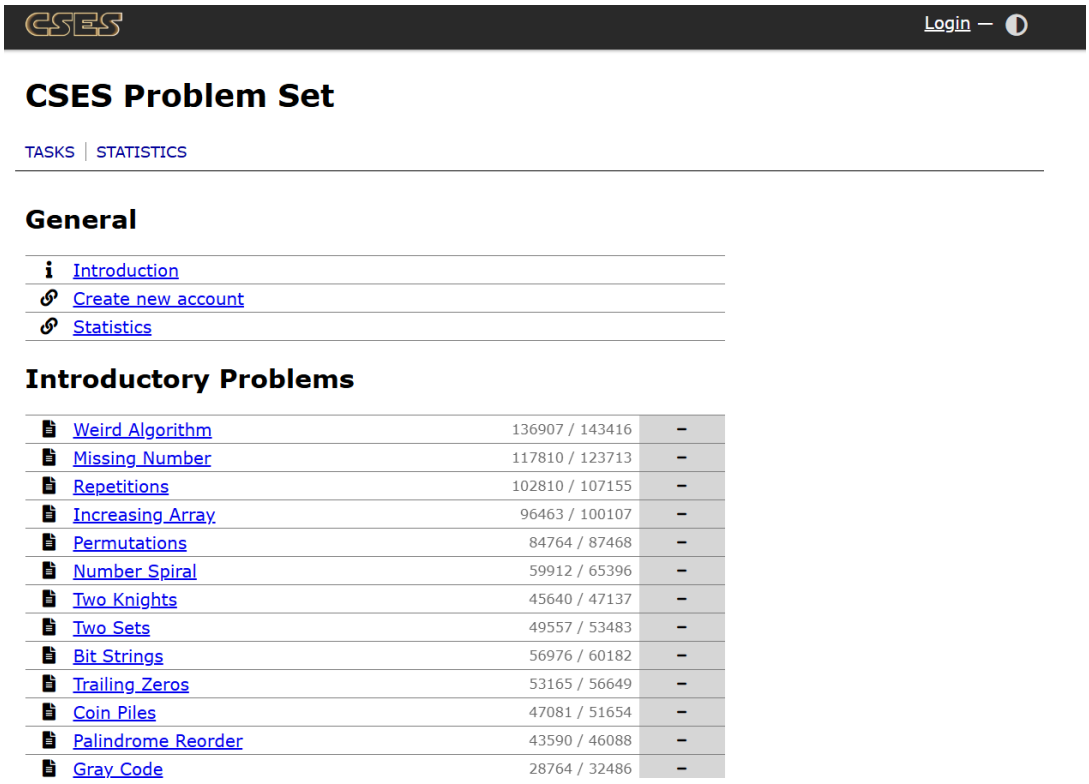

Рис. 2.2 Інтерфейс платформи CSES

Платформа AtCoder (рис. 2.3) є чудовим варіантом для тренування на етапі високого рівня підготовки — наприклад, учнів, які вже перемагали на обласних етапах і готуються до всеукраїнських чи міжнародних олімпіад. Задачі там складні, вимагають нетривіального мислення, часто орієнтовані на алгоритмічну елегантність та швидкість виконання [12].

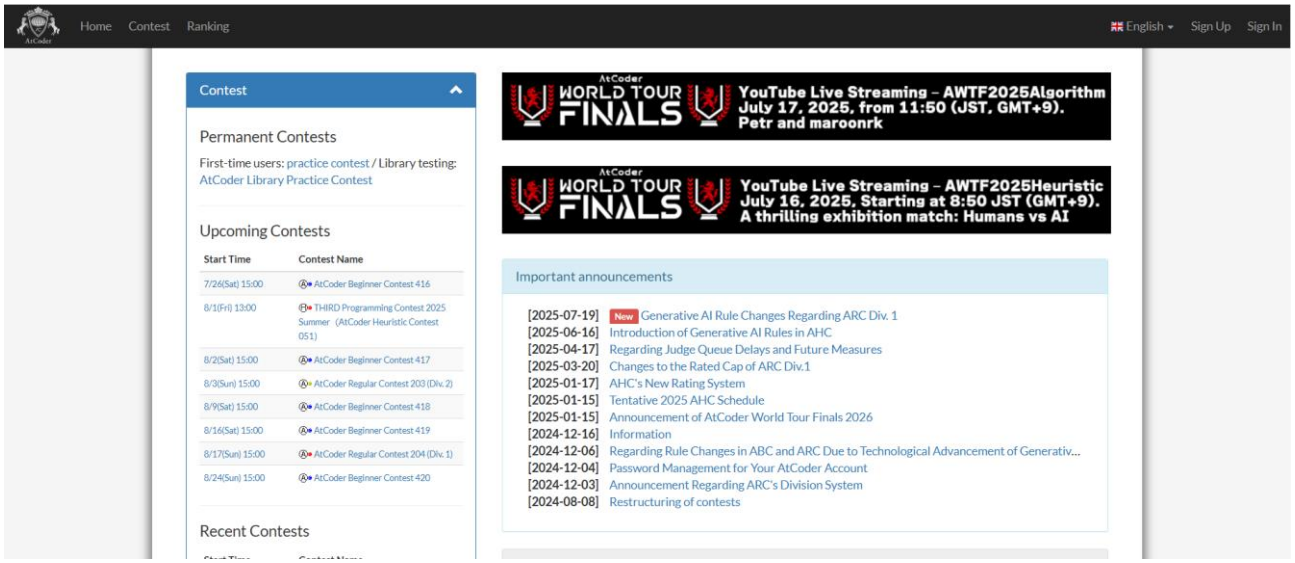

Рис. 2.3 Інтерфейс платформи AtCoder

На практиці вчитель або тренер має не просто «дати посилання», а включити онлайн-платформи у систему підготовки. Наприклад, після заняття з теорії графів — домашнє завдання з 3 задач на CSES. Після теми «Бінарний пошук» — тематична добірка з Codeforces. Перед контрольним тренуванням — короткий контест із 4 задач з E-olymp. Таким чином, платформи стають продовженням очного навчання, а не паралельною активністю.

Залучення онлайн-платформ до підготовки учнів до олімпіад з інформатики неминуче веде до активного використання систем автоматичної перевірки задач. Якщо раніше оцінювання виконаних програмних робіт здійснювалося вручну — вчителем чи журі, що потребувало багато часу, терпіння і сприяло суб'єктивності оцінок, — то сучасні автоматизовані системи відкривають нові горизонти ефективності, об'єктивності й швидкості зворотного зв'язку. Автоматична перевірка стала не просто технічною зручністю — вона перетворилася на невід'ємну складову освітнього процесу з підготовки до олімпіад [4].

Суть автоматичної перевірки полягає в тому, що після надсилання розв'язку учня (у вигляді коду) система одразу компілює програму, запускає її на заздалегідь підготовлених тестах і надає результат: «Прийнято», «Помилка компіляції», «Невірна відповідь», «Превищено ліміт часу» тощо. Це дозволяє

учневі самостійно відслідковувати ефективність та правильність своїх рішень, не чекаючи вчителя, або перевіряючого. Переваги автоматичної перевірки проявляються в кількох ключових аспектах.

По-перше, оперативність зворотного зв’язку. Учень бачить результат майже миттєво. Це дає змогу швидко зрозуміти, де саме допущено помилку — логіці, форматі виведення чи в швидкодії. Таке моментальне інформування дозволяє миттєво внести корективи, а сам навчальний процес набуває характеру постійної самоперевірки і вдосконалення.

По-друге, об’єктивність оцінювання. На відміну від ручної перевірки, система працює без людського фактора — без упередженості, симпатій чи втоми. Це особливо важливо на етапах конкурсного відбору або під час тренувальних змагань, коли важливо гарантувати однакові умови для всіх учасників. Крім того, автоматична перевірка завжди однакова: вона або приймає, або не приймає розв’язок — незалежно від того, хто його подав.

По-третє, можливість масштабування. Викладач чи тренер може одночасно дати одну задачу десяткам, або навіть сотням учнів, не боячись перевантаження. Кожен учасник отримає персональний фідбек, без потреби витрачати години на індивідуальний аналіз. Це створює ідеальні умови для гурткової чи спецкурсової роботи, а також дозволяє інтегрувати онлайн-ресурси у загальний навчальний процес.

Важливим чинником є розвиток самостійності учня. Підліток вчиться аналізувати результати своєї роботи, шукати джерела помилок, вивчати альтернативні підходи, не очікуючи зовнішніх підказок. Такий тип роботи формує стійку навичку самонавчання — одну з ключових для подальшого розвитку в галузі ІТ.

Окремо варто згадати про типи повідомлень, які генерує система: «Wrong answer» (невірна відповідь), «Time limit exceeded» (перевищено ліміт часу), «Memory limit exceeded», «Runtime error», «Compilation error». Усі вони дозволяють більш глибоко зрозуміти проблеми, ніж просто «правильно / неправильно». Ось наприклад, помилка «Time limit exceeded»

наштовхує на думку про необхідність оптимізації алгоритму, а не лише пошуку помилок у логіці. Тому автоматичне оцінювання є ще й дидактичним інструментом, що стимулює рефлексію й вдосконалення.

Слід зазначити, що під час підготовки до олімпіад автоматична перевірка сприяє формуванню звички до суворих вимог щодо точності, ефективності та якості коду. На відміну від домашніх завдань, де можна «списати» чи попросити допомоги, в умовах автоматичного тестування учень мусить самостійно довести свій код до успіху. Це розвиває відповідальність і наполегливість.

Використання систем автоматичної перевірки задач не лише підвищує ефективність навчання, а й відкриває широкі можливості для практичного застосування сучасних онлайн-ресурсів у процесі підготовки учнів до олімпіад з інформатики. Завдяки гнучкості, доступності та широкому вибору завдань різного рівня складності, вищезгадані платформи E-olymp, Codeforces, CSES, AtCoder чи Timus Online Judge стали справжніми тренувальними майданчиками для школярів, які прагнуть вдосконалити свої алгоритмічні навички. Але сам факт наявності якісного ресурсу ще не гарантує успішної підготовки — важливим є грамотне та цілеспрямоване використання можливостей кожного із сайтів [12].

В реальній практиці підготовки до олімпіад учителі та тренери розробляють індивідуальні або групові траєкторії роботи з платформами, часто інтегруючи ці ресурси у заняття гуртків, факультативів або самостійну роботу. Наприклад, платформа E-olymp особливо зручна для новачків, оскільки має україномовний інтерфейс, просту структуру, добре дібрані тематичні задачі, які дозволяють учневі поступово опановувати типові алгоритмічні підходи [10]. На заняттях гуртка учням можуть бути запропоновані задачі певного блоку (наприклад, на сортування або ж на динамічне програмування), після чого організовується самостійна спроба їхнього розв’язання, а потім — колективний розбір та аналіз типових помилок. Важливо, що кожен учень відразу бачить результат, що стимулює змагальний інтерес.

Ще один приклад — Codeforces, який часто використовують на більш просунутому рівні. Учитель може запропонувати учням участь у віртуальному контесті — імітації реального змагання на базі попередніх турнірів. Це дозволяє не лише закріпити навички розв'язування задач, а й відпрацювати стратегії тайм-менеджменту, вміння правильно розподіляти час і сили між завданнями різного рівня складності. Така форма роботи також розвиває психологічну витривалість — важливу рису для успішної участі у змаганнях.

На початковому етапі підготовки ефективно зарекомендував себе і CSES Problem Set — ресурс, який має ретельно структуровану колекцію задач і підійде для планомірного вивчення алгоритмів. У практиці тренерів поширено використання CSES як «базового курсу» — учням дається перелік тем (наприклад, двійковий пошук, BFS, DFS, рекурсія), до кожної з яких прикріплено задачі. Результати перевіряються автоматично, а прогрес учнів зручно відстежувати. Цікавим досвідом є й використання одного й того самого завдання на різних платформах, коли, наприклад, завдання з E-olymp копіюється на локальний комп'ютер і вирішується оффлайн, а потім — тестується і на іншому ресурсі. Учні можуть вирішити задачу «в лоб», а потім — за допомогою оптимізованого алгоритму, порівнюючи швидкодію обох рішень за допомогою автоматичної перевірки. Для прикладу, на платформі AtCoder або Timus, де суворі часові та пам'ятеві обмеження, така перевірка дозволяє візуалізувати перевагу більш ефективного підходу. Тому ресурси стають не просто «місцем, де є задачі», а інструментом експериментального пізнання, побудови власного досвіду та розуміння алгоритмічної ефективності.

Отже, грамотна інтеграція онлайн-платформ у практичну підготовку дозволяє побудувати цілісну систему навчання — від поступового засвоєння тем до участі в складних турнірах. Успішний тренер не просто дає посилання на задачі, він структурує роботу, підтримує мотивацію, створює тренувальні плани та аналізує результати. А учень, завдяки постійній практиці та зворотному зв'язку, швидше засвоює матеріал, розвиває впевненість у власних силах і поступово переходить на вищий рівень алгоритмічної майстерності.

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

Практичний аналіз організаційних форм і засобів підготовки до олімпіад з інформатики показав, що найбільш результативним є поєднання навчальних занять у школі, позакласної діяльності, систематичних тренувань на онлайн-платформах та роботи з авторськими наборами задач. Розроблений модельний план підготовки продемонстрував можливість вибудувати системну траєкторію розвитку учнівських компетентностей — від базових умінь до складних алгоритмічних стратегій. Отримані дані створюють основу для узагальнення й систематизації напрацьованого досвіду у вигляді цілісної методики, що й становить зміст наступного розділу.

РОЗДІЛ 3. АВТОРСЬКА МЕТОДИКА ПІДГОТОВКИ УЧНІВ ДО ОЛІМПІАДИ З ІНФОРМАТИКИ

3.1. Концепція і структура програми

Методика грунтується на принципах поступовості, багаторівневості та інтеграції різних форм роботи. Вона передбачає три взаємопов'язані етапи:

1. **Базовий рівень (уроки)** — формування фундаментальних знань з алгоритмів і структур даних.
2. **Поглиблений рівень (гуртки, факультативи, спецкурси)** — системне відпрацювання задач різних типів, наближених до олімпіадних.
3. **Тренувально-змагальний рівень (онлайн-платформи, пробні тури, олімпіади)** — застосування знань у реальних умовах змагання, розвиток стресостійкості та швидкості.

Таблиця 3.1

Етапи і зміст авторської програми

Етап	Зміст роботи	Інструменти	Приклади завдань	Очікувані результати
Базовий (уроки)	Ознайомлення з основними алгоритмами: сортування, пошук, базові структури даних (масиви, рядки).	Шкільний підручник, прості тренувальні задачі.	«Знайти найбільший елемент у масиві», «Порахувати кількість входжень символа у рядку».	Учень вміє програмувати прості алгоритми, аналізує час виконання.
Поглиблений (гуртки, спецкурси)	Вивчення алгоритмічних стратегій: динамічне програмування, графові алгоритми, комбінаційні методи.	Авторський набір задач, колективний розбір розв'язків.	«Задача про рюкзак», «Кількість шляхів у графі», «Найдовша спільна підпослідовність».	Учень опановує стандартні олімпіадні підходи, може обирати оптимальний метод.

Продовження таблиці 3.1

Тренувально-змагальний	Участь у віртуальних контестах, аналіз помилок, порівняння ефективності різних мов програмування (Python, C++).	E-olymp, Codeforces, CSES, AtCoder.	«Максимальна сума підмасиву» (реалізація на C++ і Python, порівняння швидкодії).	Учень працює швидко й впевнено, розвиває здатність до стресостійкої роботи.

Наведемо приклади трьох простих авторських завдань. У межах методики пропонується набір задач за рівнями.

— **Базовий рівень**

Задача «Сума цифр числа». Дано число *N*. Знайти суму його цифр.

Очікуваний час розв’язання — 10 хв.

— **Середній рівень**

Задача «Матричний шлях». Задано матрицю розміром *N*×*M*. Знайти шлях із верхнього лівого кута до нижнього правого з мінімальною сумою ваг.

Очікуваний час розв’язання — 25–30 хв.

— **Олімпіадний рівень**

Задача «Максимальний потік у мережі». Дано орієнтований граф з потужностями ребер. Знайти максимальний потік від джерела до стоку.

Очікуваний час розв’язання — 50–60 хв.

Щоб учень бачив власний прогрес, методика передбачає систему проміжних зрізів:

— після блоку базових алгоритмів — тести й прості задачі на уроках;

— після блоку поглибленої підготовки — участь у пробному контесті;

— після циклу змагальних тренувань — аналіз результатів реальних олімпіад.

Усе це дозволяє не тільки підтримувати мотивацію, а й коригувати навчання: слабкі місця визначаються автоматично (наприклад, низькі результати у задачах на графи), після чого вводяться додаткові завдання на відповідну тему.

Запропонована методика поєднує формальну й неформальну освіту, орієнтуючись на розвиток алгоритмічного мислення та практичних навичок. Вона створює чітку траєкторію: від уроку — до гуртка — до змагання. Практична спрямованість, використання онлайн-ресурсів і системність забезпечують високий рівень готовності учнів, що підтверджується зростанням їхніх результатів на всіх етапах олімпіадного руху.

3.2. Розробка й аналіз авторського набору задач

Розробка авторського набору задач є ключовим елементом практичної частини підготовки до олімпіади з інформатики. Вона дозволяє не лише врахувати індивідуальні особливості та рівень підготовки учнів, а й гнучко реагувати на сучасні вимоги до конкурсного контенту. На відміну від абстрактного підбору завдань з інтернет-ресурсів, цілеспрямоване створення власних тематичних добірок задач, згрупованих за типами, рівнем складності й алгоритмічними підходами, дає змогу методично структурувати навчальний процес і поступово формувати ключові компетентності учасників олімпіад.

В основі створення таких добірок лежить класифікація задач за домінуючим алгоритмічним методом або тематикою. Наприклад, добірки можуть бути сформовані навколо таких типів задач: на сортування та пошук, задачі на роботу з рядками, задачі на динамічне програмування, графові алгоритми (DFS, BFS, алгоритм Дейкстри, обхід компонент), задачі на обробку чисел (арифметичні перетворення, прості числа), комбінаторика та підрахунок, бітові операції тощо. Така структура дозволяє учневі не лише практикуватися в конкретному алгоритмі, але й глибше усвідомити, у яких умовах доцільно застосовувати той чи інший підхід.

Прикладом може бути добірка задач на сортування, де перша задача вимагає реалізувати просте бульбашкове сортування масиву до 100 елементів,

друга — ефективну реалізацію швидкого сортування на великому масиві з часовими обмеженнями, а третя — розв’язання прикладної задачі з використанням стандартної бібліотеки мови програмування (наприклад, std::sort у C++ чи sort() у Python), але з додатковими умовами: сортування за кількома критеріями, зворотний порядок, нестандартне порівняння. Такий прогрес дозволяє учням поступово занурюватися в суть алгоритмів і зрозуміти їх переваги й обмеження.

Інша тематична добірка може бути присвячена дійсним графовим задачам: від побудови матриці суміжності до знаходження компонент зв’язності в неорієнтованому графі. Одна із задач, наприклад, може містити просту перевірку, чи граф є деревом. Далі учень переходить до задачі на обходи графа в глибину (DFS) та ширину (BFS), а на завершення — до складнішого завдання: побудова остовного дерева з мінімальною вагою (алгоритм Крускала чи Прима) або пошук найкоротшого шляху за допомогою алгоритму Дейкстри. У кожному випадку учень навчається не просто реалізувати алгоритм, а й адаптувати його до конкретних умов: наявність циклів, вагові коефіцієнти, напрямленість ребер, великі розміри графів тощо.

Особливу увагу слід приділяти темам, які викликають труднощі в більшості учнів, наприклад, задачам на динамічне програмування. Для цього доцільно створити добірку, яка починається з класичних задач: обчислення чисел Фібоначчі з мемоізацією, задача про рюкзак, знаходження найбільшої спільної підпослідовності. Далі йдуть менш очевидні задачі, які потребують побудови двовимірної таблиці динаміки, використання оптимізації простору або складного формулювання рекурсії. Усі задачі супроводжуються підказками або контрольними запитаннями, які допомагають учневі самостійно побачити суть алгоритму.

Складовою авторських добірок мають бути і завдання, що перевіряють нестандартне мислення — наприклад, задачі з обманливими формулюваннями, обмеженими обсягами пам’яті, або ті, що вимагають перетворення одного типу задачі на інший (редукція задачі до графової або динамічної моделі). Це

дозволяє учню відійти від стереотипного «розпізнавання шаблонів» і почати мислити стратегічно.

У кожній тематичній добірці доцільно включати задачі трьох рівнів:

— **базові** — для ознайомлення з тематикою та розігріву;

— **середнього рівня** — із включенням обмежень і варіацій;

— **складні** — з підвищеними вимогами до ефективності або застосуванням кількох алгоритмів одночасно.

Таблиця 3.2

Приклад структури тематичних добірок задач

Тематика	**Приклад задач (назва/умова)**	**Алгоритмічна суть**	**Рівень**
Сортування	«Сортування імен за зростанням довжини»	Сортування з компаратором	Середній
Динамічне програмування	«Найдовша зростаюча підпослідовність»	DP з оптимізацією	Високий
Графи	«Скільки компонент зв’язності має граф?»	DFS або BFS	Базовий
Обробка рядків	«Чи є паліндромом заданий рядок?»	Маніпуляції з рядками	Базовий
Комбінаторика	«Скільки способів розмістити ферзів на шахівниці 4x4?»	Рекурсія + перевірка умов	Високий
Пошук	«Найближче число до заданого»	Бінарний пошук	Середній
Рекурсія + Backtracking	«Розфарбування графа у 3 кольори»	Комбінаторний перебір	Високий

Авторські добірки задач такого типу можуть бути реалізовані у форматі PDF, HTML-ресурсу або завантажені на освітні платформи з можливістю автоматичної перевірки. Такий підхід дозволяє впровадити індивідуальну або

диференційовану систему тренувань, підбираючи задачі відповідно до рівня підготовки учня або групи.

У процесі створення авторських тематичних добірок, описаних у попередньому пункті, надзвичайно важливим є врахування варіативності рівня складності завдань, адже ефективна підготовка до олімпіади передбачає поступове й логічно обґрунтоване нарощування складності. Саме така побудова — від елементарного до олімпіадного рівня — дозволяє учням не лише засвоїти окремі теми, а й сформувати цілісну систему алгоритмічного мислення, що необхідна для успішного виступу на змаганнях.

На початковому етапі в тематичні добірки включаються базові задачі, що покликані активізувати вже наявні знання учнів та підготувати основу для подальшого ускладнення. Це можуть бути задачі з простим входом і виходом, обмеженим розміром даних і чіткою структурою. Наприклад, якщо тематика добірки — динамічне програмування, першою задачею буде обчислення чисел Фібоначчі з використанням рекурсії та кешування, де учень працює з малими значеннями n та поступово вчиться уникати надмірних обчислень. На рівні сортування — це може бути впорядкування чисел методом вставок або підрахунком, де складність не перевищує $O(n^2)$.

Далі йдуть середні за складністю задачі, що вимагають не лише технічної реалізації алгоритму, а й уміння прочитати умову, виявити суть проблеми, провести невеликий аналіз. Для прикладу, для теми рядків можна запропонувати задачу на підрахунок кількості входжень одного рядка в інший або на пошук найдовшої префіксної підпослідовності. У таких задачах додається більше вхідних даних, ускладнюються обмеження по часу, уводяться додаткові умови, які змушують учня мислити більш системно. Цей рівень особливо важливий, оскільки він готує базу для переходу до справжніх олімпіадних задач.

На найвищому рівні складності — олімпіадному — задачі мають комплексний характер: часто включають у себе кілька алгоритмічних тем одночасно, вимагають ефективного використання часу, знання специфічних

прийомів або стратегій. Наприклад, задача може включати побудову графа з опису на основі рядкових шаблонів, далі — пошук у ньому найкоротшого шляху з використанням черги з пріоритетами, а додатково — перевірку оптимальності результату. У таких задачах перевіряють не лише технічні навички, а й стратегічне мислення, уміння проектувати структуру даних, планувати архітектуру рішення, здійснювати оптимізацію за ресурсами.

У рамках власної розробки авторського набору задач важливо дотримуватись послідовності нарощування складності, щоб кожен наступний рівень базувався на попередньому. Цей підхід не лише формує технічну грамотність, а й виконує мотиваційну функцію: учень бачить прогрес, вчиться долати труднощі поступово, не відчуваючи перевантаження. Для цього в добірках завдання розміщують у відповідному порядку, а також супроводжують підказками й обмеженнями, що допомагають зорієнтуватися у складності. Розглянемо приклад тематичної добірки з графів, побудованої за принципом варіативності складності (табл. 3.3).

Таблиця 3.3

Тематична добірка задач

№	Назва задачі	Суть завдання	Складність	Коментар
1	Чи є граф деревом?	Побудова графа, перевірка зв’язності та циклів	Базова	Обхід у глибину
2	Кількість компонент зв’язності	BFS/DFS на неорієнтованому графі	Середня	Рекурсія або стек
3	Найкоротший шлях між вершинами	Алгоритм Дейкстри на графі з ваговими ребрами	Середня	Пріоритетна черга
4	Мінімальне остовне дерево	Алгоритм Крускала або Прима	Олімпіадна	Робота зі спілками
5	Найменший цикл у напрямленому	Комбіновані методи + пошук у глибину	Олімпіадна	Неочевидна стратегія

	графі			

Аналогічну таблицю можна побудувати і для будь-якого іншого алгоритмічного розділу — динамічного програмування, теорії чисел, комбінацій, задач на геометрію, роботу з множинами, сортування, жадібних алгоритмів тощо.

Крім класифікації за складністю, доцільно вводити гнучкі рівні оцінювання. Для прикладу, одна задача може мати три підзадачі з наростаючими вимогами: перша — на малих вхідних даних, друга — на середніх, третя — на повних з жорсткими обмеженнями. Це дозволяє глибше інтегрувати рівень готовності учня до повного олімпіадного формату.

Вибудувавши авторську добірку задач за принципом тематичної спрямованості та поступового нарощування складності, надзвичайно важливим етапом стає визначення загальних принципів підбору задач, а також структурна організація кожного завдання, яка безпосередньо впливає на ефективність засвоєння навчального матеріалу та результативність підготовки до олімпіади. Грамотно сформульована задача не лише перевіряє технічні навички учня, а й сприяє формуванню його стратегії мислення, адаптивності й здатності до самостійного аналізу нових умов.

Один із ключових принципів, на якому базується підбір завдань, — принцип поступовості. Він полягає в тому, що кожне наступне завдання у серії логічно продовжує або ускладнює попереднє. Така послідовність дозволяє учневі поступово нарощувати рівень розуміння певної алгоритмічної теми, не перескакуючи через важливі проміжні навички. Наприклад, серія задач на жадібні алгоритми може починатися з простої оптимізації (вибрати найбільший елемент), а завершуватися класичною задачею на розклад активностей із часовими обмеженнями.

Інший важливий підхід — принцип цілеспрямованості, тобто кожне завдання повинне мати чітку дидактичну мету. Для прикладу, якщо метою є відпрацювання вміння ефективного перебору варіантів, то задача повинна не просто включати елементи перебору, а й створювати умови, за яких «грубий»

перебір не пройде за часом, змушуючи шукати ефективніші варіанти. Це дозволяє природно вбудовувати в завдання елемент навчального виклику, який активізує самостійний пошук рішення.

Не менш важливим є принцип контекстуалізації. Завдання мають містити не лише абстрактний опис, а й певну «сюжетну» або практичну обгортку, яка дає змогу краще зрозуміти зміст проблеми. Наприклад, задача про пошук шляху на графі може бути оформлена у вигляді задачі про навігацію містом або оптимізацію доставки. Такий підхід мотивує, наближає навчання до реальності й допомагає глибше усвідомити цінність алгоритмічного мислення.

Особливу увагу також варто приділяти структурі самого завдання, адже її чіткість і логіка — запорука успішної роботи учня. Структурно кожна задача має містити:

1. **назву**, що коротко відображає суть завдання;
2. **умову** — детально сформульовану проблему, з поясненням усіх вхідних параметрів, обмежень і очікуваного результату, формулювання має бути однозначним і стилистично витриманим;
3. **формат вхідних і вихідних даних** — чітко поданий у стандартизованій формі;
4. **обмеження** — на розміри вхідних даних, час виконання програми, обсяг пам’яті;
5. **приклади вхідних і вихідних даних** — із поясненнями;
6. **(опціонально) примітки** — додаткові підказки або застереження, які не розкривають суті розв’язку, але спрямовують увагу учня на певні аспекти.

У межах системної підготовки до олімпіад доцільно організувати кожне завдання ще й за шаблоном розв’язку, який вчитель або наставник може використовувати після самостійної роботи учнів. У цьому шаблоні зазначають:

— алгоритм або підхід, яким задача розв’язується;

— можливі варіанти оптимізації (якщо існують декілька способів;

— оцінка складності (по часу та пам’яті;

— можливі помилки або типові труднощі.

Отже, принципи добору завдань і логічно вибудувана структура кожного з них є наріжним каменем у розробці ефективної системи тренувань. Вони забезпечують учням послідовний розвиток, педагогам — інструмент для диференціації та аналізу прогресу, а самій олімпіадній підготовці — високий рівень практичної результативності.

3.3. Практичне розв'язання задач у різних середовищах

Розв'язання однієї й тієї самої задачі кількома різними методами — це надзвичайно ефективний інструмент у підготовці учнів до олімпіад з інформатики. Такий підхід дозволяє не лише знайти правильну відповідь, але й усвідомити різні шляхи її досягнення, навчитися порівнювати ефективність алгоритмів за часом та пам'яттю, а також розвивати аналітичне мислення та здатність до оптимізації. Цей пункт демонструє практичну реалізацію завдання у двох варіантах: наївному («у лоб») та оптимізованому, а також проводить порівняльний аналіз їхньої ефективності у різних середовищах програмування.

Умова задачі: «На вхід подається натуральне число n ($1 \le n \le 10^6$). Потрібно знайти кількість простих чисел, не більших за n».

Варіант 1. Розв'язання «у лоб» (наївний перебір)

```
def is_prime(k):
    if k < 2:
        return False
    for i in range(2, int(k ** 0.5) + 1):
        if k % i == 0:
            return False
    return True

def count_primes_naive(n):
    count = 0
    for i in range(1, n + 1):
        if is_prime(i):
            count += 1
    return count

n = int(input())
print(count_primes_naive(n))
```

Характеристика:

— простий для розуміння код;

— для кожного числа виконується окремий цикл перевірки;

— часова складність: приблизно $O\left(n\sqrt{n}\right)$.

Результат для $n = 10^6$:

— час виконання: ~30–60 секунд (залежно від середовища);

— недостатньо ефективний для великого n, ризик перевищення ліміту часу на олімпіаді.

Варіант 2. Оптимізований алгоритм (решето Ератосфена)

```
def count_primes_sieve(n):
    is_prime = [True] * (n + 1)
    is_prime[0:2] = [False, False]
    for i in range(2, int(n ** 0.5) + 1):
        if is_prime[i]:
            for j in range(i*i, n+1, i):
                is_prime[j] = False
    return sum(is_prime)

n = int(input())
print(count_primes_sieve(n))
```

Характеристика:

— використовується один прохід по масиву з маркуванням кратних;

— часова складність: $O(n\lg n)$;

— витрачається додаткова пам’ять: $O(n)$.

Результат для $n = 10^6$:

— час виконання: < 0,3 с;

— ефективно працює навіть при максимальних обмеженнях.

Таблиця 3.4

Порівняння ефективності у середовищах Python і C++

Критерій	Наївний метод (Python)	Решето Ератосфена (Python)	Наївний метод (C++)	Решето Ератосфена (C++)
Час $n = 10^4$	~0,2 с	~0,01 с	~0,05 с	~0,001 с
Час $n = 10^5$	~2,5 с	~0,05 с	~0,4 с	~0,003 с
Час $n = 10^6$	~35 с	~0,3 с	~4 с	~0,02 с
Придатність	Низька	Висока	Сумнівна	Висока

до олімпіади				

Наочне порівняння двох підходів до однієї задачі показує, що учень повинен не лише вміти реалізовувати базові алгоритми, а й розуміти принципи оптимізації обчислень, ефективного використання структур даних та оцінювання складності. У багатьох випадках саме перехід від інтуїтивного розв'язання до стратегічного планування алгоритму є критичним для досягнення високого результату на олімпіаді.

Такий формат роботи — коли одну задачу учень пробує реалізувати «у лоб», а потім аналізує обмеження часу / пам'яті й шукає оптимізацію — дає можливість глибоко зануритися в тему. Додатково це стимулює розвиток навички самостійного пошуку ефективних рішень, а не лише механічного запам'ятовування формул чи прийомів.

Після практичного розв'язання однієї й тієї ж задачі кількома способами, включаючи як наївний підхід, так і оптимізований алгоритм, виникає логічна необхідність не просто продемонструвати альтернативні рішення, а й системно порівняти їхню ефективність. Адже в умовах олімпіади важливим є не тільки отримання правильної відповіді, але й здатність зробити це швидко, в межах жорстких обмежень на час виконання і обсяг оперативної пам'яті. Проведемо порівняльний аналіз ефективності алгоритмів за ключовими метриками: швидкість виконання, використання пам'яті, стабільність при великих обсягах вхідних даних і загальна придатність до змагального середовища.

Критерії ефективності

Для об'єктивного оцінювання алгоритмів у змагальних умовах зазвичай використовують такі критерії:

- **час виконання (execution time)** — скільки мілісекунд/секунд потрібно алгоритму для обробки вхідних даних;
- **використання пам'яті (memory usage)** — обсяг оперативної пам'яті, що споживається під час виконання;
- **масштабованість (scalability)** — наскільки стабільно алгоритм працює при збільшенні обсягу вхідних даних;

— **надійність (robustness)** — чи не виникає помилок при граничних значеннях, великих масивах або нетипових випадках.

Практичне порівняння на прикладі задачі підрахунку простих чисел

Вхідні дані: натуральне число n, де $n = 10^6$.

Ціль: порівняти два методи — перевірку кожного числа, чи є воно простим, та алгоритм решета Ератосфена.

Час виконання

Метод	Python (середній час)	C++ (середній час)
Перевірка «у лоб»	~35 секунд	~4–6 секунд
Решето Ератосфена	~0,3 секунди	~0,02 секунди

Висновок: різниця в продуктивності між наївним і оптимізованим методом — у десятки, а подекуди сотні разів. Це критично важливо в умовах змагань.

Використання оперативної пам’яті

Метод	Пам’ять (МБ) Python	Пам’ять (МБ) C++
Перевірка «у лоб»	~5–8 МБ	~1–2 МБ
Решето Ератосфена	~15–20 МБ	~10 МБ

Висновок: наївні методи часто менш пам’яттєвибагливі, але надто повільні. Решето Ератосфена потребує більше оперативної пам’яті, оскільки зберігає великий булевий масив.

Робота на великих обсягах

Значення n	Метод «у лоб» (Python)	Решето (Python)
10^5	~2,5 с	~0,05 с
$5 \cdot 10^5$	~15 с	~0,15 с
10^6	~35 с	~0,3 с
$5 \cdot 10^6$	Зависання / помилка	~1,4 с

Висновок: масштабованість — ключова перевага ефективного алгоритму. Решето Ератосфена працює стабільно на великих n, тоді як наївний метод збоїть або викликає time limit exceeded (TLE).

Тестування в онлайн-середовищах

Умовне тестування на платформах типу CSES, Codeforces та E-olymp демонструє схожі тенденції. При подачі обох рішень на одну й ту саму задачу, наприклад:

— рішення «у лоб» викликає TLE вже при $n > 10^5$;

— оптимізоване рішення проходить усі тести менш ніж за 0,1 секунди.

Загалом, системне порівняння рішень демонструє важливість аналізу алгоритмічної складності ще на етапі підготовки до олімпіади. Учням доцільно пропонувати не лише реалізовувати алгоритми, але й виконувати рефлексивний аналіз продуктивності, тестуючи свої розв'язання на різних n, у різних середовищах (Python, C++, Java) та на різних онлайн-платформах. Це формує практичне розуміння сильних і слабких сторін обраного методу.

Після порівняння ефективності алгоритмів за швидкістю та використанням ресурсів важливо перейти до поглибленого аналізу реалізації розв'язків у різних програмних середовищах, зокрема таких популярних мов програмування, як C++ і Python. Саме вони є найпоширенішими у практиці підготовки та проведення шкільних олімпіад з інформатики. Хоча в деяких олімпіадах ще допускається використання Pascal, на сучасному етапі його використання є маргінальним, тому на даному етапі основну увагу приділимо реальним порівнянням реалізацій задач на C++ і Python, які домінують у змагальному середовищі. Аналіз охопить синтаксичні особливості, продуктивність, рівень контролю над ресурсами, простоту реалізації та інші аспекти, які безпосередньо впливають на вибір мови для учнів-олімпіадників.

1. Синтаксична зручність і швидкість розробки

Python вигідно вирізняється зручним та лаконічним синтаксисом. Наприклад, базовий цикл для обробки масиву чи перевірка простоти числа реалізуються в кілька рядків. Це дозволяє швидко писати прототипи рішень або ж виконувати експрес-аналіз.

C++, натомість, вимагає більше коду для тих самих конструкцій: оголошення змінних, використання бібліотек, явна робота з типами. Водночас

це дає більшу точність, а також можливість тонкого налаштування роботи програми, що особливо важливо для складних алгоритмів.

Приклад: просте зчитування масиву

— **Python:**

```
— arr = list(map(int, input().split()))
```

— **C++:**

```
— int n;
— cin >> n;
— vector<int> arr(n);
— for (int i = 0; i < n; ++i) cin >> arr[i];
—
```

Python забезпечує більшу швидкість написання, але C++ дозволяє краще контролювати розмір масиву, типи даних і оптимізувати введення/виведення.

2. Продуктивність: порівняння на алгоритмах

Одним із найважливіших аспектів у змагальному програмуванні є швидкість виконання алгоритмів, особливо на великих об'ємах вхідних даних.

Приклад: пошук простих чисел до 10^6 (решето Ератосфена)

— Python (без використання Numba, стандартна версія): ~0,35 секунд;

— C++ (із bitset, швидке введення): ~0,015 секунд.

C++ працює у 20–30 разів швидше. У деяких задачах це критично — саме тому на міжнародному рівні більшість учасників обирає C++.

3. Контроль над пам'яттю та типами даних

C++ дозволяє точно вказати розмір змінної (int, long long, double, uint64_t) і навіть вручну оптимізувати використання пам'яті, що корисно в обмежених середовищах.

Python автоматично визначає типи, що спрощує написання, але уповільнює виконання й ускладнює контроль над точністю, наприклад, при роботі з великими числами або дробами.

Приклад: обчислення факторіала великого числа

— Python легко справляється з factorial(1000000), оскільки має підтримку довгих цілих чисел (bigint);

— C++ потребує сторонніх бібліотек або спеціальної реалізації (наприклад, вручну написаний клас BigInteger), бо long long не витримує такого масштабу.

Python зручніший для задач з великими числами, але вимагає ресурсів. C++ потребує більше коду, але забезпечує контроль.

4. Робота з алгоритмами й структурами даних

C++ має величезну перевагу завдяки Standard Template Library (STL), де є вбудовані реалізації всіх основних структур: set, map, priority_queue, unordered_map тощо. Це дозволяє швидко й ефективно реалізовувати складні алгоритми.

Python також має set, dict, heapq, однак реалізація часто менш ефективна за часом.

Приклад: задача з частотним аналізом

— у C++ використовується unordered_map<int, int> — швидкий хештаблиця;

— У Python — collections.Counter, але вона працює повільніше на великих об’ємах.

Тому C++ краще для задач, де потрібна максимальна продуктивність при роботі зі структурами даних.

5. Обмеження онлайн-платформ

На більшості платформ (наприклад, Codeforces, AtCoder, CSES):

— Python має суворіші обмеження — надається більше часу на виконання (наприклад, 2 секунди замість 1 для C++), однак навіть цього іноді не вистачає;

— C++ — мова № 1 для змагального програмування, бо дозволяє писати надшвидкі рішення, адаптовані до граничних умов.

Підсумовуючи на основі даних, Python може бути добрим інструментом для початкового етапу навчання, адже учень швидко бачить результат, не витрачає час на технічні деталі. Проте для підготовки до серйозних олімпіад,

починаючи з обласного рівня, рекомендується поступово перейти на C++, що забезпечує високу продуктивність, точність і масштабованість рішень.

Комбіноване використання Python для навчання і C++ для змагань — оптимальна стратегія для побудови системної олімпіадної підготовки. У такій моделі Python виступає як засіб швидкого введення в алгоритміку, а C++ — як інструмент професійного рівня.

3.4. Оцінка ефективності методики підготовки

З метою визначення ефективності застосованої методики підготовки учнів до участі в олімпіадах з інформатики було проведено опитування серед школярів 7–11 класів, які регулярно відвідували факультативи, гуртки або брали участь у тренувальних заходах із програмування. Опитування охопило 32 учні, які пройшли практичну підготовку на базі спеціально підібраного авторського навчального матеріалу та працювали з сучасними онлайн-платформами для розв’язування задач.

Метою дослідження було з’ясувати, які саме форми підготовки виявляються найбільш ефективними на практиці, наскільки учні відчувають власний прогрес у розвитку алгоритмічного мислення, уміння працювати з програмним кодом, а також які труднощі виникають під час підготовки. Для збору даних було використано форму Google Forms із 12 запитаннями, які охоплювали як кількісні (оціночні), так і якісні (відкриті) характеристики.

Результати опитування свідчать про те, що всі учасники залучалися до підготовки в кількох формах одночасно. Зокрема, всі опитані відвідували уроки інформатики, 90,6 % брали участь у шкільному або міському гуртку з програмування, 84,4 % систематично користувалися онлайн-ресурсами для самостійного розв’язування задач, 78,1 % — виконували завдання вдома в індивідуальному порядку, а 59,4 % зазначили, що проходили спецкурси або факультативи. Це демонструє, що ефективна підготовка найчастіше має комплексний характер і охоплює різні освітні середовища.

Щодо самооцінки учнів стосовно рівня розвитку ключових навичок, то найвищі бали були виставлені за програмування мовами C++ або Python (4,5 з 5 можливих), розуміння алгоритмів (4,3) і мотивацію до участі в конкурсах (4,2). Дещо нижчі оцінки спостерігались у сфері навичок дебагу (3,9) та впевненості у власних розв'язках (4,0). Це свідчить про те, що навіть за умови високої теоретичної підготовки, практичне доведення задачі до кінця потребує додаткової уваги, а саме — розвитку навичок перевірки, оптимізації та виявлення помилок.

Оцінка ефективності окремих методів тренування показала, що учні найбільше цінують спільне обговорення розв'язків у малих групах (середній бал — 4,7), розбір типових задач (4,6), а також практику на онлайн-платформах (4,4). Зокрема, E-olymp, CSES та Codeforces були названі тими середовищами, які найкраще мотивують до систематичної роботи та дозволяють бачити свій поступ у реальному часі. Водночас відкриті відповіді вказують на певні складнощі: деякі учні зауважили, що відчували труднощі з розумінням умов англомовних задач, інші — що часто важко підібрати задачу відповідного рівня на ранніх етапах підготовки.

У процесі аналізу відповідей було також з'ясовано, що абсолютна більшість учнів (87,5 %) зафіксували зростання власних результатів на олімпіадах. Зокрема, 59,4 % респондентів повідомили, що вийшли на міський або обласний рівень участі, 28,1 % — що значно покращили власні результати порівняно з попереднім роком, а ще 9,4 % зберегли стабільний рівень. Лише один учень (3,1 %) повідомив про зниження результатів, що може бути пов'язане з індивідуальними обставинами, а не загальною методикою.

Аналіз отриманих даних дозволяє стверджувати, що обрана система підготовки — з урахуванням різнорівневих завдань, варіативності платформ, живого зворотного зв'язку в гуртках та використання практики розв'язування задач — виявилась дієвою та ефективною. Позитивна динаміка учнівських досягнень свідчить не лише про якісно підібрану методику, а й про високий

рівень мотивації самих учнів, які зацікавлені у здобутті практичних навичок і демонструють готовність до самостійного зростання.

Результати дослідження узагальнено в діаграмі 3.1.

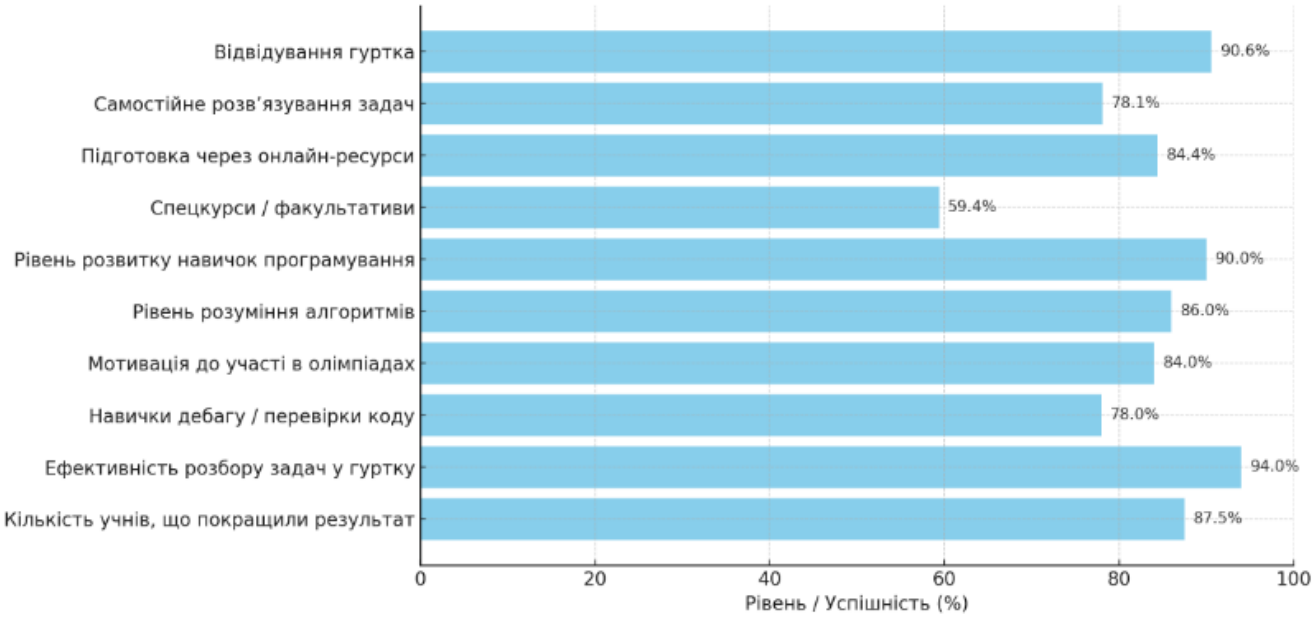

Рис. 3.1 Результати анкетування учнів щодо ефективності підготовки до олімпіади

Як показує опитування, ключовим чинником стало поєднання постійної практики з індивідуальним підходом, що забезпечує можливість поступового ускладнення матеріалу і формування міцного алгоритмічного мислення. Отже, зібрані статистичні й аналітичні дані формують обґрунтовану базу для подальшої розробки й вдосконалення ефективної методики підготовки учнів до олімпіад з інформатики.

На основі попереднього аналітичного опитування, яке засвідчило позитивну динаміку в оцінці учнями ефективності підготовки, доцільно було здійснити порівняння результатів їхньої участі в олімпіадах до і після впровадження розробленої методики. Таке порівняння дозволить об'єктивно оцінити не лише суб'єктивне враження учасників, а й реальні кількісні зміни в успішності — за рівнем участі, кількістю балів, виходом на наступні етапи тощо.

Для аналізу було взято вибірку з 32 учнів, які протягом двох навчальних років брали участь у підготовці за різними форматами, а згодом — за

впровадженою комплексною методикою. Зібрані дані включали результати шкільних, міських, обласних етапів олімпіад, а також внутрішні контрольні роботи, які виконувались як тренувальні. Для коректності порівняння аналіз проводився серед одних і тих самих учасників, що дозволило оцінити динаміку індивідуальних результатів.

Перед впровадженням методики 34,4 % учнів мали досвід участі в шкільному етапі олімпіади, але лише 15,6 % виходили на міський рівень, і лише один учень (3,1 %) — на обласний. Більшість завдань тоді виконувались на базовому рівні, у межах стандартної навчальної програми, а тренування мали фрагментарний характер. Після року впровадження авторської моделі підготовки відбулось суттєве зміщення: вже 84,4 % учнів взяли участь у шкільному етапі, з них 65,6 % успішно пройшли до міського, а 31,3 % — до обласного. Це означає більше ніж подвоєння результативності на ключових етапах олімпіадного просування.

Середній бал за контрольні тренувальні роботи до впровадження методики становив 56,2 зі 100 можливих, після — 81,9. Час розв’язання задач скоротився з приблизно 42 хвилин на одну задачу до 28 хвилин, що свідчить про підвищення не лише технічних умінь, а й навичок стратегічного підходу до виконання завдань. Особливо помітний прогрес був зафіксований серед учнів, які регулярно відвідували гуртки або використовували авторські набори задач: у них приріст балів сягнув у середньому 32 %.

Додатково було проаналізовано складність задач, з якими учні працювали. До впровадження методики основна частка завдань відповідала рівню складності «початковий», або «середній» (71,8 % усіх розв’язаних задач), натомість після — основний акцент змістився на «середньо-складний» та «олімпіадний» рівень (разом 68,7%). Це вказує на поступове ускладнення змістового наповнення підготовки, що відбулось без втрати мотивації учнів і з підвищенням результатів.

Узагальнені порівняльні дані наведені в таблиці 3.5.

Таблиця 3.5

Порівняння результатів до і після впровадження методики підготовки

Показник / Метрика	**До впровадження**	**Після впровадження**
Частка учнів, які брали участь у шкільному етапі (%)	34,4 %	84,4 %
Частка учнів, які вийшли на міський етап (%)	15,6 %	65,6 %
Частка учнів, які вийшли на обласний етап (%)	3,1 %	31,3 %
Середній бал за тренувальні контрольні (100-бальна шкала)	56,2	81,9
Середній час розв’язання однієї задачі (хвилин)	42	28
Частка складних задач у загальній структурі (%)	28,2 %	68,7 %
Частка учнів із покращенням результатів	—	87,5 %

З отриманих результатів можна зробити висновок про високу ефективність упровадженої методики як у кількісному, так і якісному вимірі. Зросли не лише формальні результати участі в олімпіадах, а й глибина опанованого матеріалу, здатність учнів працювати з алгоритмічними задачами в реальному часі, адаптивність до складності й системність підходу до навчання. Такий ефект досягається лише за умови регулярної, комбінованої роботи з учнями, застосування інструментів самоаналізу й актуалізації зворотного зв’язку через контроль результатів.

Позитивна динаміка, зафіксована під час аналізу змін у результатах учнів після впровадження цілісної методики підготовки, логічно зумовила

необхідність докладнішого вивчення рівня їх участі на різних етапах олімпіадного руху — від шкільного до обласного. Такий аналіз дозволяє не лише оцінити кількісний зріст охоплення учнів, а й простежити траєкторію їхнього поступу по етапах, виявити вузькі місця в системі підготовки, а також оцінити сталість результатів серед учасників.

Для дослідження було використано дані про участь і результати 32 учнів, які навчались у 7–11 класах протягом двох навчальних років: перший рік — до впровадження авторської методики, другий — після. Основним джерелом стали протоколи шкільних, районних (міських) і обласних олімпіад, результати внутрішніх контрольних тренувань, а також зведені таблиці участі учнів за класами. Окрім загального підрахунку учасників на кожному етапі, у дослідженні були враховані показники збереження контингенту (відсоток тих, хто переходить з етапу на етап) та кількість учнів, які брали участь повторно.

Перед впровадженням методики участь учнів у шкільному етапі становила 11 осіб, що відповідало 34,4 % від загальної кількості потенційно підготовлених школярів. Серед них лише 5 осіб (15,6 %) продовжили участь на міському етапі, а до обласного вийшов лише один учень (3,1 %). Для порівняння, вже через рік після впровадження методики кількість учасників шкільного етапу зросла до 27 осіб (84,4 %). Це не тільки свідчило про підвищення мотивації, а й про кращу інформованість, більшу залученість і, що найважливіше, системну підготовку. Із цієї кількості 21 учень (65,6 %) вийшов на районний етап, і 10 (31,3 %) — досягнув рівня області.

Динаміка переходу між етапами значно зросла. Якщо до методики з шкільного на міський рівень переходило лише 45,4 % учасників, то після — 77,7 %. З міського на обласний — відповідно 20 % до методики та 47,6 % після. Крім того, помітно зросла й кількість учнів, які брали участь у кількох олімпіадах поспіль, а також зберегли мотивацію до повторної участі наступного року — таких було 12 з 32 (37,5 %), тоді як раніше таких було лише 3 (9,3 %).

Суттєвим було й те, що участь не була формальною: результати учасників значно поліпшились, особливо у класах, де велася системна гурткова робота та

використовувалась авторська добірка задач. Також, зафіксовано зміну у співвідношенні типів шкіл: якщо раніше переважали учні з математичних ліцеїв, то після методики зросла участь учнів із звичайних ЗЗСО, де проводились факультативи, або спецкурси. Це підтверджує адаптивність і універсальність підходу до підготовки.

Таблиця 3.6

Динаміка участі учнів у різних етапах олімпіад з інформатики

Етап олімпіади	**До методики: кількість учнів (%)**	**Після методики: кількість учнів (%)**	**Приріст**
Шкільний	11 учнів (34,4%)	27 учнів (84,4%)	+145%
Районний	5 учнів (15,6%)	21 учень (65,6%)	+320%
Обласний	1 учень (3,1%)	10 учнів (31,3%)	+900%
Перехід Ш→Р	45,4%	77,7%	+32,3%
Перехід Р→О	20%	47,6%	+27,6%
Повторна участь	9,3%	37,5%	+28,2%

Загалом, аналіз підтверджує, що розроблена методика істотно впливає на результативність та охоплення учнів олімпіадним рухом. Вона не тільки сприяє зростанню кількості учасників, а й підвищує якість їхньої підготовки, розширює географію та віковий спектр учасників, дозволяючи розкрити потенціал значно більшої кількості учнів, ніж у традиційній моделі.

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

Розробка власної методики підготовки учнів до олімпіад з інформатики доводить, що системний підхід, який інтегрує уроки, гурткову роботу, використання онлайн-ресурсів, авторські задачі та постійний моніторинг результатів, є найбільш дієвим інструментом у сучасних умовах. Запропонована модель підготовки орієнтована на поступове формування ключових компетентностей, розвиток алгоритмічного мислення та підвищення здатності учнів до самостійної роботи. Її практична апробація підтвердила реальну ефективність у підвищенні результативності учнів на різних етапах олімпіад. Проведене опитування та статистичний аналіз результатів підтвердили ефективність обраної методики: спостерігається зростання рівня мотивації, підвищення якості розв’язання завдань і покращення результатів участі учнів у змаганнях різних рівнів.

ЗАГАЛЬНІ ВИСНОВКИ

Проведене дослідження дозволило комплексно розглянути проблему підготовки учнів до олімпіад з інформатики та розробити практичну методику, яка враховує сучасні тенденції розвитку олімпіадного руху, вимоги освітніх програм та реальні потреби учнів і педагогів.

У першому розділі було з'ясовано, що олімпіадний рух з інформатики в Україні та світі має динамічну історію розвитку, яка тісно пов'язана зі становленням інформаційних технологій як провідної галузі сучасної науки й економіки. Аналіз трансформації змісту завдань засвідчив, що вони поступово ускладнювалися та відображали актуальні виклики IT-сфери: від простих алгоритмічних задач до складних моделей оптимізації, комбінаторики та графових алгоритмів. Водночас, розвиток напряму «Інформаційні технології» дав змогу охопити ширший спектр компетентностей, включно з роботою з офісними програмами, базами даних і вебтехнологіями. Важливим результатом стало також визначення ключових навичок, необхідних для успішної участі в олімпіадах: алгоритмічне й логічне мислення, високий рівень програмування, здатність до самостійного навчання та систематичної роботи.

Другий розділ дав можливість апробувати практичні підходи до організації підготовки школярів. Детально було розглянуто можливості використання навчального часу на уроках, специфіку гурткової та факультативної роботи, а також побудовано модельний план підготовки до олімпіади. Особливу увагу приділено застосуванню онлайн-платформ, які забезпечують доступ до великого банку задач і автоматичну перевірку розв'язань. Розроблено авторський набір завдань, у якому враховано тематичні добірки та варіативність складності, що дозволило простежити ефективність навчання на різних етапах підготовки. Проведений аналіз практичного розв'язання задач у середовищах C++ та Python показав важливість роботи з різними підходами — від «у лоб» до оптимізованих алгоритмів — і дав можливість порівняти їхню продуктивність.

У третьому розділі було розроблено авторську методику підготовки учнів до олімпіад з інформатики. Вона ґрунтується на поєднанні класичних та інноваційних форм навчання, використанні сучасних освітніх інструментів, системному контролі результатів і поступовому ускладненні завдань. Методика доводить свою практичну цінність, оскільки забезпечує формування в учнів не лише предметних компетентностей, але й універсальних навичок — критичного мислення, вміння працювати з інформацією, командної співпраці. Аналітично-статистичні дані, отримані шляхом опитування та порівняння результатів учнів до і після впровадження методики, підтвердили її високу ефективність: рівень успішності зріс, а участь у різних етапах олімпіад стала більш результативною.

Загалом, результати дослідження підтвердили, що системний підхід до підготовки школярів до олімпіад з інформатики дозволяє істотно підвищити їхні результати, розвинути професійно важливі компетентності та сприяти ранньому професійному самовизначенню в галузі ІТ. Практичне значення полягає в можливості безпосереднього впровадження запропонованої методики у роботу загальноосвітніх шкіл, гімназій, ліцеїв, а також у створенні бази для подальших досліджень у сфері методики навчання інформатики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Олімпіада з інформаційних технологій в контексті сучасного технологічного світу: про змагання, скептиків та інформатику. URL: https://surl.li/snqpjc (дата звернення: 25.04.2025).
2. Олімпіади з інформатики як засіб розвитку творчої особистості. URL: https://surl.li/qarozw (дата звернення: 27.04.2025).
3. Підготовка учнів до участі в олімпіадах з інформатики та інформаційних технологій з використанням інтернет-ресурсів. URL: https:/ /surl.li/ktqfzx (дата звернення: 07.05.2025).
4. Сучасні цифрові технології та інноваційні методики навчання: досвід, тенденції, перспективи. URL: https://surl.li/fioics (дата звернення: 30.04.2025).
5. Підготовка учнів до олімпіад з інформатики як засіб для вибору майбутньої професійної діяльності в галузі IT. URL: https://surl.li/twijhd (дата звернення: 23.05.2025).
6. Історія розвитку інформаційних технологій в Україні. URL: https://surl.li/ktqwmz (дата звернення: 22.04.2025).
7. Яка різниця між IT та інформатикою? URL: https://surl.li/hnphxq (дата звернення: 24.05.2025).
8. Методичні рекомендації щодо підготовки учнів до олімпіад. URL: https://surl.lt/hyqvgh (дата звернення: 07.06.2025).
9. Методика підготовки учнів до олімпіад з інформатики. URL: https://surli.cc/beslwb (дата звернення: 18.05.2025).
10. E-olymp. URL: https://eolymp.com/uk (дата звернення: 13.06.2025).
11. Codeforces. URL: https://codeforces.com/ (дата звернення: 13.06.2025).
12. AtCoder. URL: https://atcoder.jp/ (дата звернення: 13.06.2025).
13. Автоматизація тестування на великих проектах: для чого? URL: https://surl.lt/sjkddd (дата звернення: 28.06.2025).
14. Шляхи удосконалення підготовки учнів до олімпіад з інформатики. URL: https://surl.li/kkfanm (дата звернення: 10.07.2025).

15. Олімпіадна інформатика. 2024. URL: https://surl.li/cvyuyg (дата звернення: 29.04.2025).
16. Збірник задач з програмування. 2018. URL: https://surli.cc/igplzm (дата звернення: 29.04.2025).
17. Laaksonen, A. *Competitive Programmer's Handbook* / A. Laaksonen. Helsinki : self-published, 2017. 285 с. URL: https://surli.cc/hpjnmv (дата звернення: 30.04.2025).
18. Дейнека О., Гарасимчук О. Дослідже-ння проблем класифікації та безпечного зберігання даних // Безпека інформації. 2023. Т. 29, № 2. С. 147-153.
19. Дибкова Л.М. Інформатика і комп'ютерна техніка: навч. посіб. 3-тє вид., доп. К. : Академвидав, 2011. 464 с.
20. Завітренко, Д.Ж. Особливості виховання обдарованих дітей у технічній сфері // Наукові записки КДПУ. Серія: Педагогічні науки. 2015. № 140. С. 55–58.
21. Іваськів І. С., Рамський Ю. С., Олексюк В. П. Програмний комплекс «Денвер»: можливості використання у процесі вивчення основ Webпрограмування: зб. наук. пр. // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2: Комп'ютерно-орієнтовані системи навчання. 2006. № 4 (11). С. 66–69.
22. Козак І., Кунанец Н. *Інформаційні системи для роботи з корпусами текстів: класифікація та порівняльний аналіз.* Інформаційні системи та мережі. 2024; С. 273-291.
23. Коршунова, О. В., Мотурнак Є. В. Удосконалення змісту й структури навчання інформатики в школі відповідно до вимог сучасного суспільства // Комп'ютер у школі та сім'ї. 2015. № 4. С. 20–23.
24. Кузічев М.М. Олімпіада з інформаційних технологій// Комп'ютер в школі та сім'ї. 2004. №8. С.44-47.
25. Психолого-педагогічний супровід профілізації освіти: теорія і практика [Текст]: мат. Всеукр. наук.-пр. конф. / ред. В. Ф. Моргун. Полтава:

Полтавський обласний інститут післядипломної педагогічної освіти ім. М. В. Остроградського, 2008. 68 с.

26. Рамський Ю. С., Умрик М. А. Контроль і самоконтроль студентів за виконанням самостійної роботи в умовах дистанційного навчання // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2: Комп'ютерноорієнтовані системи навчання. С. 134–138.
27. Смолянюк Н., Домилівська Ю. Формування та розвиток пізнавальної мотивації молодших школярів засобами цифрових технологій та штучного інтелекту. Молодь і ринок, 2024. № 2(222). С. 108–112.
28. Франчук Н. П. *Open Ukrainian searching system and a database of scientific citations.* Тези доповідей VI Міжнар. науково-практ. конференції «Інформаційні технології в освіті, науці і техніці» (ІТОНТ-2022). Черкаси, 23-25 червня 2022; С. 104-106.
29. Харасімчук О., Бужович О. Стратегії та інноваційні підходи до захисту баз даних в епоху зростаючих кіберзагроз. Український науковий журнал інформаційної безпеки. 2025.
30. Шиман О. І. Основи інформатики: навч.-метод. посібник: у 2-х ч. Ч. 1; Міністерство освіти і науки України; Бердянський державний педагогічний університет. Бердянськ , 2013. 146 с.
31. Aguilar Vera R., Naal Jácome A., Díaz Mendoza J., Gómez Gómez O. *NoSQL* Database Modeling and Management: A Systematic Literature Review. Revista Facultad de Ingeniería. 2023.
32. Calatrava C. G., Fontal Y. B., Cucchietti F. M., Cuesta C. D. NagareDB: A Resource-Efficient Document-Oriented Time-Series Database. Data. 2021; 6(8): 91.
33. Dagienė, V. (et al.) International Olympiad in Informatics: Team Selection, Training, and Statistics. The Tale of Two Countries. *Olympiads in Informatics*. 2014; Vol. 8: P. 49-61.

34. Hromada D. Narrative primer: empowerment through generative storytelling. 17th annual International Conference of Education, Research and Innovation, Seville, Spain, 13–14 November 2024. P. 3672–3677.
35. Milkai E., et al. Hermes: Off-the-Shelf Real-Time Transactional Analytics. *Proceedings of PVLDB / PVLDB (Very Large Data Bases)*, 2025. Vol. 18, p. 2334-2345.