МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Фізико-математичний факультет
Кафедра комп'ютерних наук та інформаційних технологій

Реєстраційний №_____
Дата реєстрації_____

МЕТОДИЧНІ АСПЕКТИ ВИВЧЕННЯ РОЗДІЛУ «АЛГОРИТМИ ТА ПРОГРАМИ» ПІД ЧАС ДИСТАНЦІЙНОГО НАВЧАННЯ

Кваліфікаційна (дипломна) робота здобувача вищої освіти
другого (магістерського) рівня вищої освіти
спеціальності 014 Середня освіта
предметної спеціальності 014.09 Середня освіта (Інформатика)
освітньої програми «Інформатика в закладах освіти»
25Мд-СОінф групи
Стельмашенко Яніни Андріївни
Науковий керівник:
кандидат педагогічних наук, доцент
Кривонос Олександр Миколайович

Рекомендована до захисту
рішенням кафедри комп'ютерних наук та інформаційних технологій

Протокол №6 від «10» листопада 2023р.

Зав.кафедри _____ _____Олена УСАТА_____
(підпис) (Ім'я Прізвище)

Житомир – 2023

Дата захисту _____

Результат захисту

оцінка за національною шкалою	кількість балів за 100 бальною шкалою	ECTS

Голова ЕК _____ _____
(підпис) (Ім'я Прізвище)

Члени ЕК _____ _____
(підпис) (Ім'я Прізвище)

_____ _____
(підпис) (Ім'я Прізвище)

_____ _____
(підпис) (Ім'я Прізвище)

_____ _____
(підпис) (Ім'я Прізвище)

Секретар ЕК _____ _____
(підпис) (Ім'я Прізвище)

ЗМІСТ

ВСТУП

Актуальність. Вже багато років базовий курс інформатики в школі готує спеціалістів та просто обізнаних в інформаційно-комунікаційних технологіях людей. Тенденція обрання професії пов’язаної з ІТ-сферою повністю паралельна тенденції зростання даної ланки праці на ринку, адже жодна прогресивна компанія не може не мати корпоративного веб-сайту, жодна компанія не може не використовувати базу для зберігання та обробки даних та жодне підприємство не обходиться без автоматизації простих та складних процесів. Тому при такому попиті на фахівців у сфері ІТ, і народжуються навчальні програми для профільного рівня з вивчення інформатики.

Але те, що здобувачі освіти навчаються інформатиці з поглибленим вивченням, зовсім не означає що їх майбутня професія – фахівець у ІТ-сфері: кожна з професій не залишається повністю відстороненою від інформаційно-комунікаційних технологій, тому варто розумітись в загальних засадах функціонування цього середовища.

У часи коли постала гостра необхідність у віддаленому навчанні, на допомогу прийшли інформаційно-комунікаційні технології. Під час дистанційного навчання вчителям потрібно максимально заохочувати, мотивувати здобувачів до освітнього процесу.

Розділ «Алгоритми» є дуже важливим адже алгоритми є фундаментальними для інформатики, вони представляють собою послідовність інструкцій, які визначають спосіб вирішення конкретної задачі, а розробка алгоритмів є важливою складовою створення програм, інформаційних систем та комп'ютерних додатків. Проте насамперед вивчення алгоритмів надає знання та навички для розв'язання різних задач і проблем, алгоритми допомагають структурувати та оптимізувати процеси вирішення завдань в різних сферах, включаючи наукові дослідження, бізнес, інженерію тощо.

З огляду на це, можна сказати що тема «Методичні аспекти вивчення розділу «алгоритми» під час дистанційного навчання» - необхідна для розгляду, а вивчення розділу «Алгоритми» є актуальним для будь-якого здобувача загальної середньої освіти.

Мета – дослідити структуру розділу «Алгоритми» та методичні аспекти його вивчення під час дистанційного навчання.

Для досягнення даної мети потрібно поставити такі **завдання**:

1. Дослідити структуру розділу «Алгоритми».
2. Дослідити поняття дистанційного навчання.
3. Розглянути методичні аспекти вивчення розділу «Алгоритми» під час дистанційного навчання.
4. Розробити приклади занять під час вивчення розділу «Алгоритми» під час дистанційного навчання.

Об'єкт дослідження - методика вивчення розділу «Алгоритми» шкільного курсу інформатики.

Предмет дослідження – елементи методичної системи, що сприяють опануванню учнями розділу «Алгоритми» шкільного курсу інформатики.

Методи дослідження:

- порівняльний,
- експериментальний,
- структурний,
- описовий.

Апробація роботи: участь у IV Всеукраїнському відкритому науково-практичному онлайн-форумі «ІННОВАЦІЙНІ ТРАНСФОРМАЦІЇ В СУЧАСНІЙ ОСВІТІ: ВИКЛИКИ, РЕАЛІЇ, СТРАТЕГІЇ» (27 жовтня 2022 р.), участь у VII Міжнародній науково-практичній студентській конференції «Професійна іншомовна підготовка в умовах глобальних комунікативних потреб» (3 листопада 2022р.)., участь у VII Всеукраїнській науково-практичній конференції з міжнародною участю "Сучасні інформаційні

технології в освіті та науці" (17-18 листопада 2022 р.) , Публікація у періодичному науковому виданні «АКАДЕМІЧНІ ВІЗІЇ».

Публікація: Результати дослідження висвітлено у 3 - матеріали конференцій та у фахових виданнях.

Структура і обсяг роботи: робота складається зі вступу, двох розділів, висновків, списку використаних джерел та літератури. Загальний обсяг – 59 сторінок, основна частина розміщена на 52 сторінках. Загальна кількість використаних джерел – 32. Кількість рисунків – 10. Таблиць у документі – 2.

РОЗДІЛ 1. РОЗДІЛ «АЛГОРИТМИ», ЙОГО СТРУКТУРА ТА ОСОБЛИВОСТІ

1.1. Структура курсу «Алгоритми»

Шкільний курс інформатики в старшій школі побудований з метою продовження формування у учнів інформаційної культури і грамотності. Оскільки інформатика в 10-11 класах є продовженням шкільного курсу, який вивчався раніше, учні, у процесі навчання, мають розширити знання з деяких тем, що вивчались в молодшій та середній школах, а також засвоїти нові знання, що допоможуть у формуванні інформаційної культури та базової компетентності у інформаційно-комунікаційних технологіях.

Проте, оскільки шкільний курс може формуватись по різному (учні починають вчити інформатику в 2, або в 5 класі), у старшій школі з'являється розгалуження у рівнях навчання: у профільному рівні діти поглиблено вивчають деякі теми, та захоплюють більше тем, у рівні стандарту, відповідно, теми вивчаються на базовому рівні.

Рівень стандарту, за навчальним планом, розрахований на 105 годин, при цьому 35 з них складає модуль, що є обов'язковим для вивчення, а решта годин займає вибірковий блок з переліком тем, які учні можуть обрати за власними вподобаннями.

Профільний рівень. Порівняно з програмою рівня стандарту, у профільних класах за мету ставиться не лише розвинути загальне розуміння ІКТ та сформувати комп'ютерну грамотність. За мету ставиться навчити більш розширеним процесам :

- розвинути логічне, аналітичне мислення, а також основні види розумової діяльності, вміння використовувати індукцію, дедукцію, аналіз, синтез, робити висновки та узагальнення;
- розвинути теоретичну базу, що відноситься до процесів перетворення, передавання та використання інформації;

• розвити вміння розв’язувати задачі різної складності за допомогою ІКТ;

• підготувати учнів до олімпіад, конкурсів, науково-дослідницьких робіт тощо;

• вивчення інформатики на творчому рівні;

• створити зв’язки між інформатикою та іншими предметами, що є у учнів.[25]

Вчителям, які навчають дітей по програмі профільного рівня рекомендується використовувати різноманітні форми організації навчання для більш ефективного навчання, що дозволить мотивувати учнів та зацікавлювати у навчальному процесі.

Основною відмінністю від стандартного рівня є те, що в профільному вчитель самостійно визначає кількість навчальних годин на вивчення тем, а також їх порядок. Це є великою перевагою, адже завдяки цьому можна орієнтуватись на профіль закладу освіти, підлаштовуватись під учнів, проте зміст робочої програми змінювати не можна .[24]

Розділ «Алгоритми» вивчається у старшій школі з профільним рівнем, таким чином для даного курсу можна взяти за основу підручник автора В. Руденко «Інформатика. Профільний рівень». Даний підручний випущений у 2019 році та рекомендований для вивчення Міністерством освіти і науки України.

Даний розділ є другим у порядку вивчення та включає шість тем:

1. Алгоритми і числа.

У темі вивчаються методи проектування алгоритмів та їх класифікація (рис.1.1): за ступенем автоматизації проектування алгоритмів і програм поділяються на неавтоматизовані (традиційні) та автоматизовані (case – технології), за методологією проектування програмних продуктів поділяються на структурне програмування та об’єктно-орієнтоване програмування.

Рисунок 1.1 Класифікація методів проектування алгоритмів

При цьому також розглядаються типові методи структурного проектування програмних продуктів: спадний метод - полягає у розбитті основної задачі на послідовні підзадачі, які виконуються крок за кроком. Кожна з цих підзадач може бути розкрита на менші, прості операції або процедури, що сприяють вирішенню загальної задачі; висхідний - вже існуючі та передбачено розроблені алгоритми для розв'язання окремих часткових завдань поступово поєднуються в єдину структуру до моменту, коли досягається розв'язання основної задачі; модульний - модуль представляє собою самостійний блок алгоритму з власною назвою, функціональною цілісністю та завершеністю. Для звернення до модуля використовується його ім'я, і виклик або оновлення модуля можливе лише через його заголовок. Один з переваг модульного підходу полягає в тому, що різні модулі можуть бути розроблені різними спеціалістами одночасно.

Кожен модуль може бути протестований і налагоджений незалежно від інших.

У підручнику розглядаються поняття кодування і складності алгоритмів. «Кодування алгоритму — це запис алгоритму мовою програмування.» [30] Складність алгоритму оцінюється витратами людських ресурсів на розробку та тестування, а також вимогами до обчислювальних ресурсів.

З практичної точки зору обчислювальні ресурси визначаються двома основними критеріями: обсягом пам'яті, необхідним для реалізації алгоритму, та часом, необхідним для виконання алгоритму. Кількість часу, яка потрібна для виконання програми, називається обчислювальною складністю. Розглядається також «основні поняття теорії чисел»: системи числення (десяткова, двійкова, шістнадцяткова) та їх переведення; «факторизація чисел» (тобто його розкладання на добуток простих множників).

2. Алгоритми сортування і пошуку даних.

У поданій темі розкривається сутність алгоритму сортування вибором, алгоритму сортування методом обміну, сортування встановленням, сортування злиттям та сортування підрахунком:

Алгоритми сортування використовуються для організації даних в певному порядку. Ось короткий огляд п'яти різних алгоритмів сортування.

Сортування вибором (Selection Sort).

Цей алгоритм вибирає найменший (або найбільший, залежно від порядку сортування) елемент і переміщує його на початок масиву. Потім алгоритм продовжує вибирати наступний найменший елемент і розміщати його за вже відсортованими елементами.

Масив розділяється на дві частини - сортовану і несортовану. Спочатку сортована частина порожня, а несортована містить всі елементи. Далі алгоритм переглядає несортовану частину масиву, щоб знайти найменший елемент. Знайдений мінімальний елемент обмінюється з першим елементом в несортованій частині масиву. Таким чином, цей мінімальний елемент стає

першим елементом сортованої частини. Діапазон сортування збільшується на один елемент, і процес повторюється. Основною перевагою сортування вибором є його простота реалізації. Однак цей алгоритм не є найбільш ефективним для великих масивів даних, оскільки він має квадратичну часову складність (O(n^2)). Тобто, час сортування збільшується квадратично зі збільшенням розміру масиву. Для великих масивів краще використовувати більш ефективні алгоритми сортування, такі як Швидке сортування (Quick Sort) чи Злиття (Merge Sort).

Сортування методом обміну (Bubble Sort).

У цьому алгоритмі порівнюються пари сусідніх елементів і обмінюються, якщо вони знаходяться у невірному порядку. Алгоритм продовжує цей процес до тих пір, поки не буде досягнуто повного сортування.

Алгоритм починається з порівняння першого та другого елементів у масиві. Якщо перший елемент більший (або менший, залежно від напрямку сортування), то вони обмінюються місцями. Після першого проходження найбільший (або найменший) елемент знаходиться в кінці масиву. Тепер алгоритм переходить до наступної пари сусідніх елементів і виконує порівняння та обмін, якщо це необхідно. Цей процес повторюється до тих пір, поки не буде досягнуто повного сортування. Кожен новий прохід по масиву призводить до того, що найбільший (найменший) елемент виходить на своє місце, тобто на останню (першу) позицію. Для покращення продуктивності може бути введена змінна, яка відстежує, чи були обміни під час проходження масиву. Якщо під час проходження не було жодного обміну, то алгоритм припиняє роботу, оскільки масив вже відсортований. Основною перевагою сортування методом обміну є його простота і легкість реалізації. Проте він має дуже високу часову складність, особливо для великих масивів даних. Його середня та найгірша часова складність становлять O(n^2), де n - розмір масиву. Тому для великих наборів даних

зазвичай рекомендується використовувати більш ефективні алгоритми сортування, такі як Швидке сортування чи Злиття.

Сортування встановленням (Insertion Sort).

Цей алгоритм розглядає кожен елемент масиву і вставляє його в потрібне місце вже відсортованої частини масиву. Процес повторюється для кожного елемента.

Масив розділяється на дві частини: відсортовану і несортовану. Спочатку відсортована частина містить лише перший елемент, а решта масиву вважається несортованою. Далі відбувається перебір елементів несортованої частини масиву, починаючи з другого елемента. Для кожного елемента визначається його правильне місце відносно відсортованої частини шляхом порівняння з елементами вже відсортованої частини. Елемент вставляється в визначене місце, а інші елементи відсортованої частини масиву зсуваються праворуч, щоб звільнити місце для нового елемента. Цей процес повторюється для кожного елемента несортованої частини масиву. По мірі виконання кожного кроку відсортована частина зростає, а несортована скорочується.

Основною перевагою сортування встановленням є його простота та легкість реалізації. Він ефективний для невеликих масивів або масивів, в яких вже велика частина елементів відсортована. Однак у випадку великих масивів цей алгоритм має квадратичну часову складність (O(n^2)), що робить його неефективним для великих обсягів даних. Для таких випадків рекомендується використовувати більш ефективні алгоритми сортування, такі як Швидке сортування чи Злиття.

Сортування злиттям (Merge Sort).

Метод сортування, який використовує стратегію "розділ і побороли". Масив розділяється на дві частини, які сортуються окремо, а потім об'єднуються відсортовані частини в один масив.

Перший крок - вихідний масив розбивається на дві рівні частини (або на більше частин у випадку багатошарового злиття). Ця операція виконується рекурсивно до тих пір, поки не залишиться лише один елемент в кожному підмасиві, оскільки один елемент вважається відсортованим. Два підмасиви злітаються шляхом порівняння елементів. Підмасиви злиттяться у великий підмасив, в якому елементи впорядковані відповідно до потрібного порядку сортування. Під час злиття, вибирається завжди найменший елемент із двох підмасивів. Цей процес повторюється для кожного рівня рекурсивного розділення, доки всі підмасиви не будуть злиті у великий відсортований масив. Після завершення останнього злиття весь масив вважається відсортованим.

Основною перевагою сортування злиттям є його стабільність і найгірший випадок часової складності. Він має часову складність O(n*log n), де n - розмір масиву, що робить його дуже ефективним для великих масивів даних. Також цей алгоритм легко паралелізується, що дозволяє його використовувати для обробки великих обсягів даних на сучасних багатоядерних процесорах.

Сортування підрахунком (Counting Sort).

Цей алгоритм підраховує кількість входжень кожного елемента в масиві і використовує цю інформацію для сортування. Він спеціально ефективний для сортування цілих чисел або інших обмежених діапазонів значень.

Спочатку алгоритм переглядає вхідний масив і підраховує кількість кожного унікального елемента в масиві. Ця інформація використовується для створення масиву підрахунків, де кожен елемент відображає кількість елементів, що збігаються. За допомогою масиву підрахунків обчислюється кумулятивна сума кількостей елементів. Ця сума покаже позиції, на які потрібно вставити кожен унікальний елемент в відсортованому масиві. Створюється новий відсортований масив. Для кожного елемента в оригінальному масиві алгоритм використовує кумулятивну суму для визначення його позиції в відсортованому масиві, вставляючи його на

правильну позицію. Після вставки кожного елемента в відсортований масив, відповідний лічильник в масиві підрахунків зменшується на одиницю. Після того як всі елементи вставлені у відсортований масив, сортування завершено.

Сортування підрахунком має часову складність O(n + k), де n - розмір вхідного масиву, а k - розмір діапазону унікальних значень. Оскільки цей алгоритм не використовує порівняння між елементами, він може бути дуже ефективним для великих масивів даних, особливо коли k (розмір діапазону) набагато менший за n. Однак він не підходить для сортування масивів з великим діапазоном значень, оскільки вимагає великого обсягу пам'яті для масиву підрахунків.

Кожен з цих алгоритмів має свої переваги та обмеження, і їх вибір залежить від конкретних завдань і вимог до ефективності сортування.

Розкривається, також, поняття послідовного пошуку, бінарного пошуку, пошуку максимального і мінімального елементів у масиві, поняття про пошук із поверненням і тернарний пошук.

3. Обробка рядків.

Рядок є одним з основних типів убудованих у мову Python об’єктів, які мають загальну назву послідовність. Рядки можуть застосовуватися для подання символів, слів, фрагментів тексту. Вони можуть також використовуватися для збереження двійкових значень байтів.

У цьому параграфі розглядаються основні відомості про рядки й операції над ними, функції і методи опрацювання рядків (такі, як len(), str(), chr(код символу), ord(символ)) та методи роботи з рядками (join (), upper (), lower (), capitalize (), find (), index (), rfind (), count (), replace ()), у параграфі наводяться приклади програм обробки рядків.

5. Графи.

Граф (рис.1.2) - це сукупність елементів і всіх відносин між цими елементами. У контексті графів, об'єкти, які представляють окремі елементи,

називаються вершинами графу, а зв'язки або відносини між цими об'єктами називаються ребрами графу.

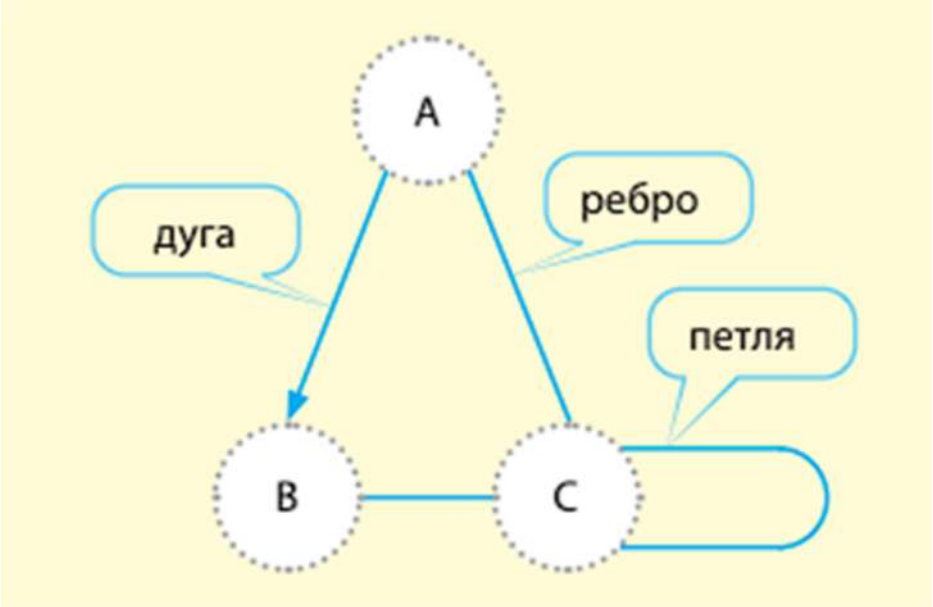


Рисунок 1.2 Найпростіший граф

Зазвичай графи відображаються за допомогою геометричних фігур, де вершини графу представлені як точки, а ребра, які з'єднують ці вершини, представлені лініями. У деяких випадках, коли важливий напрямок зв'язку, використовують стрілки для позначення напрямку. Наприклад, в односторонніх вулицях міста можуть бути показані як стрілки.

Якщо напрям зв'язку не важливий, то такі лінії називають ребрами, і граф у цьому випадку називається неорієнтованим.

Циклом називають шлях з однієї вершини графу в ту саму вершину. Циклів у графі для кожної вершини може бути декілька.

Довжиною циклу називають кількість ребер у циклі.

Граф, який не має жодного циклу, називають деревом.

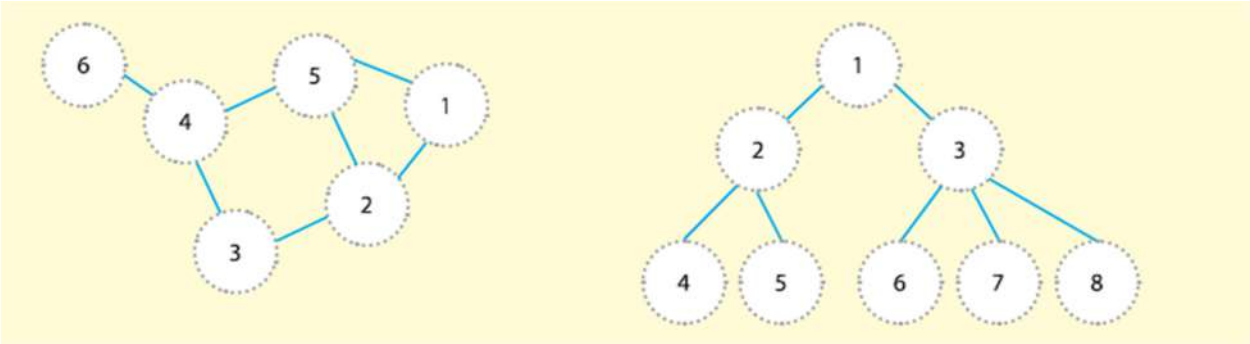


Рисунок 1.3 Графи (1- граф із циклами, 2 -граф-дерево)

Існує кілька методів, які можна використовувати для представлення графів на комп'ютері, і два з них включають в себе використання матриць суміжності і списків суміжних вершин.

Подання графів за допомогою матриць суміжності полягає в тому, що граф із n вершинами представляється у вигляді двовимірного масиву розмірністю n*n.

Під час обробки графів часто потрібно відвідати всі його вершини. Існують різні алгоритми для обходу вершин графа, разглядатимуться два з них: алгоритм пошуку в глибину та алгоритм пошуку в ширину.

Алгоритм пошуку в глибину вимагає використання двох структур даних: стеку для збереження вершин, які ще не були обробленими, і списку для збереження вершин, які вже були обробленими.

Суть алгоритму пошуку в ширину – спочатку опрацьовуються всі вершини, суміжні з поточною, а потім — «нащадки». Замість стека для збереження ще не опрацьованих вершин використовується черга.

У тему, також розглядається процес визначення найкоротшого шляху в графі, алгоритм Дейкстри, алгоритм Флойда — Уоршелла.

6. Динамічне програмування і жадібні алгоритми.

Динамічне програмування - це стратегія вирішення завдань, які можна розкласти на менші схожі підзадачі. Цей підхід застосовується при створенні програм для вирішення завдань оптимізації, де процес пошуку рішення може бути розглянутий як послідовність кроків. Основна ідея полягає в тому, що на кожному кроці процесу знаходиться оптимальне рішення, що дозволяє розбити велику та складну задачу на послідовність менших і більш простих задач.

Жадібні алгоритми - це загальна назва методів оптимізації, де на кожному етапі вибирається локально оптимальне рішення. Ці алгоритми використовуються для розв'язання задач оптимізації, які можуть бути розбиті на окремі прості підзадачі, аналогічно до того, як це робиться в алгоритмах динамічного програмування.

7. Основи обчислювальної геометрії.

Обчислювальна геометрія — це галузь комп’ютерних наук, присвячена вивченню алгоритмів розв’язування геометричних задач. У розділі розглядаються основні поняття: вектор, довжина (модуль) вектора, колінеарні вектори тощо.

Розглядаються операції над векторами:

а) Додавання та віднімання векторів. Два вектори однакового розміру можуть бути додані разом(відняті один від одного), де кожний елемент одного вектора додається до (віднімається від) відповідного елемента іншого вектора. Наприклад, вектори A і B, результатом додавання буде новий вектор C, де кожний елемент C[i] = A[i] + B[i] і при відніманні вектори A і B, результатом віднімання буде новий вектор C, де C[i] = A[i] - B[i].

б) Множення на скаляр. Вектор може бути помножений на скаляр (одиничне число), де кожний елемент вектора множиться на це число. Якщо вектор A і скаляр k, то результатом буде новий вектор B, де B[i] = k * A[i].

в) Дотичний (скалярний) добуток. Скалярний добуток двох векторів розраховується шляхом множення відповідних елементів двох векторів та

підсумовування результатів. Наприклад, вектори A і B, скалярний добуток (A x B) розраховується як A[0] * B[0] + A[1] * B[1] + ... + A[n] * B[n], де n - розмір векторів.

г) Векторний добуток (в деяких випадках). Векторний добуток двох векторів дає вектор, який перпендикулярний до площини, утвореної вихідними векторами. Ця операція не виконується для всіх видів векторів і зазвичай застосовується в тривимірному просторі.

Також досліджується обчислення площі многокутника, побудова опуклої оболонки.

1.2. Поняття дистанційного навчання.

Під час пандемії COVID-19, багато шкіл перейшли на дистанційне навчання для забезпечення безпеки учнів і вчителів. Цей перехід розширив використання віртуальних навчальних платформ, відеоконференцій та інших інструментів для навчання. Коли поняття дистанційного навчання почало з’являтись в мережі, з’явилось багато труднощів у розумінні, що таке дистанційне навчання. Дистанційне навчання – це сукупність технологій, що забезпечують надання студентам основного обсягу навчального матеріалу з використанням телекомунікаційних засобів, інтерактивної взаємодія студентів і викладачів у процесі навчання, надання учням можливості самостійної роботи з навчальними матеріалами.

Були різноманітні інструментальні спроби покращити дистанційну освіту шляхом полегшення дистанційного навчання студентів.

Найважливішою характеристикою електронного навчання є його здатність трансформуватися в нові контексти.

У дорослих людей є багато причин навчатись на відстані: обмеження в часі, відстані та фінансуванні, а також неможливість контактувати з іншими людьми як, наприклад, у пандемію. Тому, в результаті дистанційного навчання люди отримують не тільки нові знання, а і нові соціальні навички – здібність спілкуватись і співпрацювати з широким колом людей. А от причиною дистанційного навчання у дітей можуть бути лише вагомі події,

при яких навчання у очному форматі не можливе, адже попри те, що основним завданням школи є надати знання, проте очне навчання допомагає соціалізуватись, розвинути багато навичок, що важко розвинути сидячи вдома. Саме тому, перейшовши на дистанційне навчання, вчитель має заохочувати та мотивувати до навчання.

Проте є і переваги електронного навчання – гнучкість та зручність для людини, полегшення спілкування між учнями, різноманіття досвіду для учня з використання мультимедіа та невербальної подачі матеріалу. Взаємодія між вчителем та учнем забезпечує візуальне та аудіо навчання, яким учень може керувати (передивлятись лекцію).

Дистанційне навчання або дистанційна освіта — це сфера освіти, яка зосереджена на педагогіці/андрагогіці, технології та дизайні системи навчання, які ефективно включені в надання освіти студентам, викладач і студенти можуть спілкуватися асинхронно та синхронно.

Електронне навчання природно підходить для дистанційного та гнучкого навчання, але також може використовуватися в поєднанні з очним навчанням, у цьому випадку зазвичай використовується термін змішане навчання.

Електронне навчання також може стосуватися освітніх веб-сайтів, таких як ті, що пропонують робочі аркуші та інтерактивні вправи для дітей. Термін також широко використовується в бізнес-секторі, де він, як правило, стосується економічно ефективного онлайн-навчання.

Електронне навчання – це використання технологій для допомоги та покращення навчання. Зосереджуючись на використанні Інтернету в електронному навчанні, з’явилися три основні сфери використання. Це технологія для доставки, підтримки та покращення викладання та навчання.

Аналізуючи літературу можна натикнутись на список критеріїв, як забезпечується дистанційне навчання [6]:

1. Відокремлення вчителя від учня (принаймні протягом більшої частини навчального процесу).

2. Вплив освітньої організації (включаючи забезпечення оцінювання студентів).

3. Використання освітніх медіа для об’єднання вчителя та учня (і перенесення змісту курсу).

4. Забезпечення двостороннього спілкування (між викладачем, репетитором або освітньою агенцією та учнем).

Різні стратегії навчання характеризуються різними особливостями, наприклад, на відміну від традиційного навчання, що має такі характерні ознаки: контрольований вхід; безпосереднє навчання; висока частка відведеного часу; розклад діяльності; в стінах закладу.

У дистанційному навчанні є певні переваги: вільний доступ; зосередженість студента та самомотивація; навчання на основі ресурсів учня; гнучкий розклад.

Електронне навчання має переваги перед традиційним навчанням у класі, очевидними перевагами є гнучкість і економія коштів (які витрачаються на шлях до місця). Існують також переваги, які можуть бути неочевидними, приклад:

- учню не потрібно подорожувати, щоб відвідати курс;
- при дистанційному навчанні предмет може виявитись цікавішим;
- завдяки електронному навчанню учень отримає більше досвіду, оскільки вчиться на місці;
- учень добре навчиться використовувати онлайн-спільноти та мережу Інтернет (таким чином, електронне навчання може підтримувати «навчання через рефлексію та дискусію»);
- електронне навчання дає можливість учневі керувати своїм способом навчання та тим, як йому подобається вчитися;
- електронне навчання є дуже корисним для компаній, оскільки воно економить час і кошти як для учня, так і для працівників;
- це дешевше у виробництві;
- це самостійний темп;

• забезпечує узгоджене повідомлення;

• студент може працювати з будь-якого місця та в будь-який час;

• уникаються обмеження у розкладі.

Але варто враховувати, що такі переваги є лише у вільному дистанційному навчанні, адже під час пандемії та повномасштабної війни навчальні заклади застосовують дистанційне навчання як заміну очному. Тому всі переваги, що були вказані, анулюються – навчання здійснюється по розкладу, учні/студенти мають бути зосереджені на певному предметі у певний час в незалежності від ресурсоємності. Варто розуміти, що в буденний час дистанційне навчання скоріше не є способом спростити життя учня, а лише єдиний спосіб навчати його.

Сучасна школа ставить за мету якісне навчання, а основним завданням є виховати всебічно розвинену людину. За рахунок цього, перед вчителем постає задача не тільки змінити підхід до учнів та навчального процесу, а й вірно, цікаво та вмотивовано донести певну інформацію до учнів. Ось тут і потрібна інформаційно-цифрова компетентність.

Історія дистанційного навчання в Україні налічує кілька важливих етапів, починаючи з введення перших педагогічних технологій на віддаленій основі до сучасних віртуальних навчальних платформ та онлайн-курсів. Ось кілька ключових моментів історії дистанційного навчання в Україні:

1) Початок 20 століття. Перші спроби впровадження дистанційного навчання в Україні відзначилися на початку 20-го століття, коли з'явилися дистанційні курси для вчителів та студентів вищих навчальних закладів. Основний метод передачі знань був зв'язаний із використанням пошти та відправки навчальних матеріалів.

2) Радянська епоха. У Радянському Союзі були реалізовані спроби впровадження телевізійних і радіопередач для навчання, а також спеціалізовані дистанційні інститути для педагогічної підготовки та професійної підготовки.

3) Період незалежності. З моменту отримання Україною незалежності в 1991 році дистанційне навчання стало однією з важливих галузей освіти. Багато вищих навчальних закладів розпочали надавати дистанційні курси та програми, що надає студентам можливість отримати вищу освіту без фізичної присутності на уроках.

4) Впровадження інформаційних технологій. З поширенням Інтернету та розвитком комп'ютерної техніки в Україні, дистанційне навчання стало більш доступним та ефективним. Вищі навчальні заклади та навчальні платформи розпочали надавати онлайн-курси, відеолекції, інтерактивні завдання та інші ресурси для дистанційного навчання.

5) Сучасний стан. Сьогодні дистанційне навчання в Україні набуло широкого поширення та популярності. Вищі навчальні заклади, які надають онлайн-освіту, включають у себе державні університети, технічні та гуманітарні коледжі, інститути та приватні освітні заклади. В Україні існують різноманітні освітні платформи, які надають доступ до онлайн-курсів, вебінарів та відеолекцій. До найбільш відомих платформ входять "Prometheus," "Coursera," "edX," "Bitrix," "Moodle" та багато інших. Ці платформи співпрацюють із українськими університетами та навчальними закладами для надання різних онлайн-курсів. Багато вищих навчальних закладів в Україні пропонують дистанційну форму навчання для студентів. Це дозволяє отримати вищу освіту в режимі, зручному для студентів, навіть якщо вони знаходяться в інших містах або регіонах. Також, дистанційне навчання широко використовується для професійної підготовки та підвищення кваліфікації. Багато компаній та організацій співпрацюють із навчальними платформами для надання своїм працівникам можливості навчатися на відстані. А для забезпечення якості дистанційного навчання в Україні існують механізми оцінки та

нагляду. Університети та навчальні заклади використовують відповідні системи оцінювання та контролю за навчальним процесом. Дистанційне навчання широко використовується для професійної підготовки та підвищення кваліфікації. Багато компаній та організацій співпрацюють із навчальними платформами для надання своїм працівникам можливості навчатися на відстані.

В цілому, історія дистанційного навчання в Україні свідчить про поступовий розвиток та вдосконалення методів та технологій для забезпечення доступної та якісної освіти на віддаленій основі. Сучасне дистанційне навчання в Україні стикається із викликами, такими як забезпечення доступу до інтернету та комп'ютерів для всіх студентів і учнів, а також питаннями якості освіти та мотивації для самонавчання. Проте воно має великий потенціал для подальшого розвитку та розширення доступності освіти.

1.3. Передумови для вивчення курсу «Алгоритми» та його результати.

Для вивчення курсу "Алгоритми" і досягнення успіху в ньому важливо мати певні передумови. Важливо мати розуміння базових концепцій програмування, таких як змінні, оператори, умови, цикли та функції. Це є важливим фундаментом для вивчення алгоритмів. Але, у вивчені даного розділу йдеться не тільки про основи програмування, функції та методи. З огляду на це, здобувачі загальної середньої освіти повинні мати певну математичну підготовку: основи дискретної математики, теорії графів та алгебри.

Для вивчення обраної теми здобувач освіти має повністю володіти, набутими навичками безпосередньо у старшій школі вивченим розділом «Мова програмування та структури даних», у якому здобувачі освіти вивчають безпосередньо мову програмування Python, співпрацюють з різними середовищами програмування та основи об’єктно-орієнтованого

програмування у обраній мові програмування. Після проходження даного курсу учень має використовувати можливості середовища програмування для створення та налагодження програм, розв’язує задачі з використанням усіх базових алгоритмічних структур та їх комбінацій, використовувати змінні різних типів та обґрунтовує вибір типів даних, розробляти як консольні програми, так і програми з графічним інтерфейсом, знати і програмувати всі базові алгоритми обробки лінійних структур даних, такі як алгоритми вставки, видалення, пошуку елементів, сортування тощо, розробляти алгоритми розв’язування практичних завдань з використанням різних структур даних.[16]

У навчальній програмі профільного рівня для 10-11 класів у графі «очікувані результати навчання» сформовано перелік вмінь та навичок, що мають отримати учні у процесі вивчення розділу «Алгоритми» (рис. 1.7):

Знаннєва складова

- Знати методи проектування алгоритмів
- Знати і розуміти базові алгоритми
- Пояснювати структуру алгоритму та реалізувати його засобами мови програмування

Діяльнісна складова

- Реальзувати базові алгоритми засобами мови програмування.
- Планувати процес розв’язування задачі з використанням програмування.
- Створювати та налагоджувати програми за розробленими алгоритмами.
- Розв’язувати задачі з використанням базових алгоритмів.
- Обґрунтовувати вибір алгоритму для розв’язування задачі.

Ціннісна складова

- Оцінювати складність алгоритмів.
- Обґрунтовувати доцільність вибору певного алгоритму.
- Оцінювання практичного значення та ефективність програм, створених за базовими алгоритмами.
- Розпізнавання задачі, для розв’язання яких доцільно використовувати базові алгоритми.

Рисунок 1.4 Результати навчання розділу "Алгоритми"

1.4. Поняття програми і мови програмування.

На сьогоднішній день існує розмаїття мов програмування, які розрізняються і подібні між собою. Це пояснюється різноманітністю завдань, які можна вирішити за допомогою обчислювальної техніки.

Початкова рекомендація стосовно мови програмування часто вказує на Pascal як на одну з кращих вибірок для старту, оскільки вона спеціально створена для навчання основам програмування. Проте, Pascal застаріла мова,

яка не використовується в комерційних цілях. Тому, вчити себе програмуванню на Pascal може вимагати переходу до іншої мови з іншим синтаксисом та правилами.

Lazarus - це середовище розробки, яке використовує компілятор FreePascal та підтримує розробку сучасних віконних додатків. Воно позиціонується як сумісне з Delphi, але навчальний матеріал, написаний для Delphi, може бути не завжди застосованим. Інтерфейс також більш складний порівняно з IDE FreePascal. Використання Lazarus рекомендується тим, хто вже має певний досвід та кваліфікацію.

Використання C / C ++ як початкової мови програмування може мати свої проблеми, оскільки ці мови включають багато складних конструкцій, що можуть відлякати початківців. Проте, з правильним підходом, можна працювати на рівні, схожому до Pascal, використовуючи відповідні конструкції.

Вибір сучасних систем візуального проектування, таких як Delphi, Lazarus або Visual Studio на початковому етапі навчання програмуванню може створити проблеми через автоматичне генерування обсяжного коду, пов'язаного з роботою візуальних компонентів. Це може заважати розумінню основних програмних концепцій.

Отже, на сьогоднішній день рекомендацією є вивчення мови програмування Python, яка має багатий інструментарій та зручний синтаксис для початківців.

Мову програмування Python було створено близько у 1991 році Гвідо ван Россумом. Назву "Python" вона отримала від телесеріалу "Monty Python" ("Літаючий цирк Монті Пайтона"), а не від плазуна.

Після створення мови, Гвідо ван Россум опублікував її в Інтернеті, де спільнота програмістів приєдналася до розвитку і вдосконалення Python. Мова активно розвивається і нині. Офіційний веб-сайт Python розташований за адресою http://python.org .

Python - це універсальна інтерпретована, об'єктно-орієнтована високорівнева мова програмування для сценаріїв з динамічною семантикою.

Python славиться своїм чітким синтаксисом, що робить код легкочитаємим завдяки відступам та обмеженій кількості допоміжних елементів.

Python поширюється під ліцензією GNU General Public License.

Популярною середовище розробки для Python є програма IDLE, яка є кросс-платформовою та дозволяє виконувати стандартні завдання розробки на Python.

Мова програмування Python найкраще підходить як перша мова для вивчення програмістами-початківцями, оскільки вона має потужні інструменти, які відображають спосіб мислення людей і спосіб реалізації коду. [3] Крім того, він мінімізує додаткові ключові слова, необхідні для написання синтаксично правильної програми. Такий підхід видається більш продуктивним, ніж навчання мовам C або Java, де багато термінів і елементів пов’язані з особливостями мови, а не з реалізацією алгоритму. Крім того, викладачі понад дюжини університетів, таких як MIT, UC Berkeley, UC Davis, Sonoma State University, Washington University, University of Waterloo, Luther College, Swarthmore College тощо використовували його для викладання вступного курсу програмування. студентам відділення інформатики. [13] Сьогодні для фахівця з інформатики важливо вивчити принаймні одну мову програмування, оскільки всі інновації та технології базуються на глибокому розумінні комп’ютерів, операційної системи, програмного API або деяких периферійних пристроїв. Усі вони створені програмістами, які використовують особливий спосіб мислення. І щоб отримати такий спосіб мислення, потрібно звикнути до однієї з мов програмування та стати кваліфікованим у розробці програмного забезпечення. [14] Для будь-якої людини, яка починає вивчати програмування, важливо зосередитися на концепціях програмування, а не на специфіці мови, оскільки вони можуть бути різними для різних мов

програмування. Але Python забезпечує найвищий рівень програмування. Таким чином, студенту не потрібно думати про керування пам'яттю, яке є неминучим у C, або про ієрархію класів, якої неможливо уникнути в Java, або про типи та оголошення змінних, які існують майже в кожній мові програмування.

Найлегше на Python написати програму «Hello World». Більшість мов програмування вимагають написання багатьох специфічних методів або функцій, оголошень класів або програм тощо. Але Python дає можливість почати програмування без цих вимог. Ви можете перевірити приклади, наведені в розділі «Порівняння». Python є інтерпретованою мовою, тому за допомогою інтерпретатора командного рядка студент може легко перевірити, як працюють оператори чи функції. [3] В інтерпретатор Python має вбудований модуль довідки, який може значно покращити процес розуміння різних аспектів мови. Щоб зрозуміти мови програмування, першокурсникам (студентам, які не мають досвіду програмування) потрібно навчитися мислити як комп'ютерних, а це вимагає великих зусиль і повної зміни парадигми мислення. Реалізація коду Python досить проста, тому людина, яка пройшла курс елементарної математики, знайде такі інструменти, як «змінні» та «функції», легкими у використанні. Щоразу, коли програмісту потрібен прототип програмного забезпечення, можна використовувати Python із його багатою бібліотекою. Тоді програмне забезпечення може бути переписано на іншій мові, якщо це необхідно. Переваги Python значні, тому використання його як основної мови для вивчення програмування може значно вплинути на швидкість вивчення інформатики в цілому.[7]

Проте більшість вчителів та викладачів, надають перевагу вивченню C++. *Мова програмування C* надає модель пам'яті та обчислень, яка майже відповідає моделі більшості комп'ютерів. Крім того, він забезпечує потужні та гнучкі механізми для абстракції; тобто мовні конструкції, які дозволяють програмісту вводити та використовувати нові типи об'єктів, які відповідають концепціям програми. Таким чином, C підтримує стилі програмування, які

покладаються на досить пряме маніпулювання апаратними ресурсами для забезпечення високого рівня ефективності, а також стилі програмування більш високого рівня, які покладаються на типи, визначені користувачем, для забезпечення моделі даних і обчислень, ближчої до погляд людини на завдання, яке виконує комп'ютер. Ці високо рівневі стилі програмування часто називають абстракцією даних, об'єктно-орієнтованим програмуванням і загальним програмуванням.

Аналізуючи джерела можна натикнутись на вагому статтю « C++ or Python? Which One to Begin With: A Learners Perspective», де шляхом експерименту працівники університету в The Islamia University of Bahawalpur Pakistan (Пакистан) довели, що вибір мови програмування вивчення у вступному курсі програмування вимагає ретельності. З їх висновків стає зрозуміло, що вибір мови може вплинути на навчання, виживання та прогрес студентів у сфері комп'ютерної техніки, у центрі якої – програмування. Головне, чого студенти очікують від вступної мови, це легкість. Хоча значення C, Java та інших мов не можна не помітити, Python виглядає дуже сильним і життєздатним вибором, коли справа доходить до вступного курсу програмування. Студенти виявляють більше прогресу під час вивчення Python і вважають, що більшість функцій мови, включаючи цикли та умови, були легшими порівняно з C.

Також, у статті вказують на переваги мови Python серед інших мов:

• Python пропонує чистий та більш інтуїтивно зрозумілий синтаксис порівняно з іншими сучасними конкурентними мовами для цієї мети, такими як C та Java. Лістинг коду 1 містить приклад коду для базової арифметики та виведення на Python і C.

• Python має динамічну типізацію, що зменшує накладні витрати на нотацію та полегшує новачкові зосередитися на розв'язанні проблем, а не на мові.

• Python пропонує кращу виразність завдяки представленню потужних типів даних, таких як списки та словники, які можна легко навчати

одночасно з іншими базовими типами даних. Python також має дуже високий коефіцієнт операторів із C як посиланням.

• Інтерпретована природа Python дає миттєвий зворотний зв'язок. Програмування не тільки стає швидким, але й допомагає справлятися з помилками краще та простіше.

• Без використання зайвих дужок і крапок з комою Python забезпечує структурний дизайн, дотримуючись відступів і структурованого способу написання програм. Ця функція надає Python більше схожості з псевдокодом.

• Хоча це не має значення для програмістів-початківців, Python пропонує розширюваність мовних функцій завдяки дворівневій мовній конструкції.

• Середовище програмування за замовчуванням (IDLE) є відкритим і безкоштовним. Він також може працювати на більшості платформ.

Однак важливо пам'ятати, що головна мета навчання не полягає в освоєнні конкретної мови програмування, а в розвитку алгоритмічного мислення та ознайомленні з різними методами та стилями розв'язання завдань. Тому рекомендується дозволити учням ознайомитися з декількома мовами програмування без глибокого вивчення, зосереджуючись на вирішенні класичних алгоритмічних задач.

РОЗДІЛ 2. МЕТОДИЧНІ АСПЕКТИ ВИВЧЕННЯ РОЗДІЛУ «АЛГОРИТМИ ТА ПРОГРАМИ» ПІД ЧАС ДИСТАНЦІЙНОГО НАВЧАННЯ.

2.1. Методика вивчення розділу «Алгоритми» під час дистанційного навчання.

Методика викладання інформатики має свої основні цілі: визначення конкретних навчальних цілей для інформатики, формування відповідного змісту предмету і його місця в шкільному навчальному плані. Додатковими завданнями є розробка та пропозиція оптимальних методів та організаційних форм навчання для досягнення поставлених цілей, а також аналіз усіх ресурсів, пов'язаних із викладанням інформатики (навчальні посібники, програмне забезпечення, технічні засоби) та розробка рекомендацій щодо їх використання в практиці роботи вчителів.

Основною особливістю курсу інформатики є його зв'язок з іншими предметами, особливо з методичним циклом. Вивчення інформатики на сучасному рівні базується на знаннях з різних галузей наукового знання, таких як біологія (дослідження біологічних самоуправляючих систем, таких як людина та інші живі організми), історія та соціальні науки (дослідження суспільних соціальних систем), українська мова (граматика, синтаксис, семантика), логіка (мислення, формальні операції, істина, брехня), математика (числа, змінні, функції, множини, знаки, дії), психологія (сприйняття, мислення, комунікація) і багато інших.

Іншою особливістю викладання інформатики є її динамічний характер, що постійно змінюється як наука і як предмет навчання, її нестабільність та постійний розвиток і вдосконалення, як технічних, так і програмних засобів. У цьому контексті важливо максимально спиратися на результати загальної дидактики та методики суміжних дисциплін, таких як математика та фізика. Крім того, важливий аспект - це зв'язок предмета з використанням

комп'ютерів, які мають великий рівень самостійності в порівнянні з іншими приладами.

Подальший крок в навчанні - вивчення алгоритмізації та програмування, яке може бути розділене на два етапи: спочатку вивчають алгоритмізацію, а потім переходять до програмування. Багато навчальних програм фокусуються на алгоритмізації, оскільки не всі вчителі інформатики готові навчати програмуванню на конкретних мовах програмування.

Вивчення алгоритмізації сприяє розвитку алгоритмічного мислення у учнів, що є основою для подальшого вивчення програмування. Тому ця частина курсу інформатики є надзвичайно важливою, і вчителю необхідно приділяти особливу увагу та обережність під час її викладання. Основний зміст у цьому напрямку визначається через наступні ключові поняття в освітньому стандарті базового курсу інформатики та інформаційно-комунікаційних технологій:

- Алгоритм – послідовність команд, які керують діяльністю об'єкта, та являють собою чітке та точне завдання для виконавця.
- Властивості алгоритму – основні характеристики алгоритмів, які визначають їхню ефективність та ефективність виконання.
- Способи запису алгоритмів – техніки та засоби для подання алгоритмів на письмі.
- Виконавці алгоритмів – агенти, які виконують алгоритми, включаючи їх призначення, середовище, режим роботи та систему команд.
- Комп'ютер як формальний виконавець алгоритмів – роль комп'ютера в виконанні алгоритмів та взаємодія з ним.
- Основні алгоритмічні конструкції – ключові елементи алгоритмів, такі як послідовний виконання, гілкування та повторення.
- Розбиття задачі на підзадачі – поділ великої завдання на менші, більш керовані підзадачі.

- Алгоритми роботи з величинами – включають типи даних, процеси введення та виведення.

Вивчення алгоритмізації розпочинається зі введення поняття алгоритму, яке може бути складним для зрозуміння і потребує уваги. Учням важливо розуміти, що алгоритм завжди включає орієнтацію на виконавця алгоритму. Іншими словами, вони повинні розуміти, що алгоритм - це чіткі та точні інструкції для виконавця, які призводять від початкових даних до очікуваного результату.

Під час вивчення цього поняття вчителю слід акцентувати увагу на тому, що алгоритм завжди складається з команд, які включені в систему команд виконавця. Це дозволяє вчителям надавати учням завдання, щоб вони розуміли, що алгоритм завжди обмежений системою команд виконавця, та не передбачає виконавцеві самостійного прийняття рішень, які не передбачені алгоритмом. Також важливо розуміти, що алгоритм повинен бути зрозумілим та точним, адже він залежить від виконавця.

Вчителю рекомендується акцентувати увагу на інших важливих властивостях алгоритмів, таких як точність, яка позначає, що алгоритм має бути вірним та не залишати місця для розбіжностей у виконанні, і зрозумілість, що означає, що алгоритм повинен бути зрозумілим для виконавця, щоб той міг виконати його без непорозумінь.

Після вивчення основних понять алгоритмізації, вчителю слід перейти до практичних прикладів та завдань, які допоможуть учням зрозуміти та застосувати ці концепції. Зокрема, учні можуть виступати в ролі виконавців алгоритмів для простих завдань, таких як малювання геометричних фігур або знаходження коренів квадратного рівняння. Це дозволяє їм практично застосовувати засвоєні знання та розуміти їхнє практичне значення.

Крім того, важливо надавати учням можливість розробити власні алгоритми та розглядати різні варіанти вирішення завдань. Це допоможе розвивати їхню творчість та критичне мислення.

Загальна мета вивчення алгоритмізації полягає в розвитку учнівської здатності розробляти, розуміти та застосовувати алгоритми в різних сферах життя.

Аналізуючи пункт 1.1 можна дійти висновку, що структура розділу «Алгоритми» має таку будову (таблиця 2.1):

Таблиця 2.1 Будова розділу "Алгоритми

Назва розділу	Тема	Підтеми
Алгоритми	Алгоритми і числа	Методи проектування в подання алгоритмів
		Поняття про кодування і складність алгоритмів
		Основні поняття теорії чисел
	Алгоритми сортування і пошуку даних	Алгоритми сортування даних
		Алгоритми пошуку даних
	Обробка рядків	Основні відомості про рядки і операції над ними
		Функції і методи опрацювання рядків
		Приклади програм обробки рядків.
	Графи	Основні поняття і терміни теорії графів.
		Способи подання графів у комп’ютері
		Пошук у глибину і ширину.
		Визначення найкоротшого шляху у графі
	Динамічне програмування і жадібні алгоритми	Динамічне програмування
		Жадібні алгоритми
	Основи обчислювальної геометрії	Базові поняття.
		Операції над векторами
		Обчислення площі багатокутника.

		Побудова опуклої оболонки

Оскільки на модуль «Алгоритми» у ліцеях виділяється 35 годин можна дійти висновку, що календарно-тематичне планування [21] на період вивчення модулю може мати такий вигляд (таблиця. 2.2):

Таблиця 2.2 Тематичне планування до модулю "Алгоритми"

Тема 2. Алгоритми			
Алгоритми і числа			
1.		Інструктаж з БЖД. Методи проектування алгоритмів	
2.		Інструктаж з БЖД. Математична модель, вибір структури даних	
3.		Інструктаж з БЖД. Пошук оптимального алгоритму розв'язання	
4.		Інструктаж з БЖД. Узагальнення та аналіз екстремальних ситуацій	
5.		Інструктаж з БЖД. Оцінка та аналіз ефективності алгоритму	
6.		Інструктаж з БЖД. Планування, покрокова деталізація та представлення алгоритму	
7.		Інструктаж з БЖД. Декомпозиція задачі	
8.		Інструктаж з БЖД. Реалізація алгоритму мовою програмування	
9.		Інструктаж з БЖД. Основні поняття теорії чисел: системи числення	
10.		Інструктаж з БЖД. Робота з довгими числами	
11.		Інструктаж з БЖД. Факторизація чисел. ***Тематична атестація***.	
Алгоритми сортування і пошуку даних			
12.		Інструктаж з БЖД. Повторення вивчених структур даних. Прості змінні та масиви/списки	
13.		Інструктаж з БЖД. Стек	
14.		Інструктаж з БЖД. Черга	
15.		Інструктаж з БЖД. Дерево	
16.		Інструктаж з БЖД. Повторення вивчених методів сортування: вибором, обміном, вставкою	
17.		Інструктаж з БЖД. Алгоритми сортування. Квадратичні алгоритми сортування. Алгоритми сортування вибором.	
18.		Інструктаж з БЖД. Алгоритм сортування методом обміну	
19.		Інструктаж з БЖД. Сортування вставленням	
20.		Інструктаж з БЖД. Сортування злиттям	
21.		Інструктаж з БЖД. Сортування підрахунком	
22.		Інструктаж з БЖД. Алгоритми пошуку даних. Послідовний пошук	
23.		Інструктаж з БЖД. Бінарний пошук	
24.		Інструктаж з БЖД. Пошук максимального і мінімального елементів у масиві	
25.		Інструктаж з БЖД. Розв’язування практичних завдань	
26.		Інструктаж з БЖД. Поняття про пошук із поверненням і тернарний пошук. ***Тематична атестація***	
Обробка рядків			
27.		Інструктаж з БЖД. Обробка рядків. Основні відомості про	

		рядки й операції над ними	
28.		Інструктаж з БЖД. Функції і методи опрацювання рядків	
29.		Інструктаж з БЖД. Приклади програм обробки рядків	
Графи			
30.		Інструктаж з БЖД. Основні поняття і терміни теорії графів	
31.		Інструктаж з БЖД. Способи подання графів у комп'ютері	
32.		Інструктаж з БЖД. Повторення вивчених методів пошуку: лінійний та бінарний пошук	
33.		Інструктаж з БЖД. Рекурсивний пошук	
34.		Інструктаж з БЖД. Пошук у рядку	
35.		Інструктаж з БЖД. Пошук у ширину	
36.		Інструктаж з БЖД. Реалізація алгоритму пошуку в ширину	
37.		Інструктаж з БЖД. Пошук у глибину	
38.		Інструктаж з БЖД. Реалізація алгоритму пошуку в глибину	
39.		Інструктаж з БЖД. Визначення найкоротшого шляху у графі	
40.		Інструктаж з БЖД. Алгоритм Дейкстри	
41.		Інструктаж з БЖД. Реалізація алгоритму Дейкстри	
42.		Інструктаж з БЖД. Алгоритм Флойда-Уоршела	
43.		Інструктаж з БЖД. Реалізація алгоритму Флойда-Уоршела.	
44.		Інструктаж з БЖД. Розв'язування практичних завдань. ***Тематична атестація***	
Динамічне програмування і жадібні алгоритми			
45.		Інструктаж з БЖД. Динамічне програмування.	
46.		Інструктаж з БЖД. Жадібні алгоритми	
47.		Інструктаж з БЖД. Задача про рюкзак	
48.		Інструктаж з БЖД. Задача про коника-стрибунця	
49.		Інструктаж з БЖД. Задача про черепашку	
50.		Інструктаж з БЖД. Задача про розподіл ресурсів	
51.		Інструктаж з БЖД. Задача про оптимізацію маршруту	
52.		Інструктаж з БЖД. Найбільша спільна підпослідовність	
53.		Інструктаж з БЖД. Задача про центи	
54.		Інструктаж з БЖД. Задача про заявки	
55.		Інструктаж з БЖД. Загальна задача динамічного програмування	
56.		Інструктаж з БЖД. Критерії застосування задач динамічного програмування	
57.		Інструктаж з БЖД. Розв'язування практичних завдань. ***Тематична атестація***	
Основи обчислювальної геометрії			
58.		Інструктаж з БЖД. Базові поняття обчислювальної геометрії	
59.		Інструктаж з БЖД. Операції над векторами. Векторний добуток	
60.		Інструктаж з БЖД. Напрям повороту при переміщенні	
61.		Інструктаж з БЖД. Обчислення площі многокутника	
62.		Інструктаж з БЖД. Перетин відрізка	
63.		Інструктаж з БЖД. Визначення положення точки відносно багатокутника	
64.		Інструктаж з БЖД. Побудова опуклої оболонки	
65.		Інструктаж з БЖД. Розв'язування практичних завдань. ***Тематична атестація***	

Оскільки на інформатику з профільним рівнем у старших класах розраховано 5 годин на тиждень, а загальна кількість годин на рік – 175, то зрозуміло, що модуль «Алгоритми» займатиме 37% календарного плану.

Як видно з тематичного планування, у темі «Алгоритми і числа» передбачені лекційно-практичні види уроків, тому передбачено, що до кожного з уроків варто підготувати теоретичний матеріал. А у умовах дистанційного навчання матеріал обов’язково має бути з візуальним інтерактивним супроводом. Тому вчителю варто зосередити увагу учнів на таких ключових аспектах даної теми:

1. Основні поняття:

- Визначення алгоритму і його складових.
- Числа та їх типи у програмуванні (цілі числа, дійсні числа, тощо).

2. Логіка та структури даних:

- Логічні операції та їх застосування в алгоритмах.
- Робота зі структурами даних (масиви, списки, черги, стеки) для зберігання та обробки числової інформації.

3. Алгоритмізація:

- Побудова алгоритмів для розв'язання конкретних задач на числах (наприклад, знаходження простих чисел, факторизація, операції з числами, сортування тощо).
- Розробка та аналіз алгоритмів для покращення швидкості та ефективності обчислень.

4. Програмування:

- Використання мов програмування для реалізації алгоритмів.
- Відладка коду та тестування розроблених програм.

5. Практичні вправи та проекти:

- Завдання, що дозволяють учням застосовувати отримані знання на практиці.

- Розробка власних алгоритмів для розв'язання конкретних завдань чи задач.

У темі «Алгоритми сортування і пошуку даних» варто звернути увагу учнів на такі аспекти:

1. Основні алгоритми сортування:

- Пояснення роботи та реалізація алгоритмів сортування, таких як метод бульбашки, метод вибору, метод вставки, швидке сортування (Quick Sort), злиття (Merge Sort), тощо.
- Порівняння ефективності та швидкості роботи різних алгоритмів сортування.

2. Алгоритми пошуку даних:

- Вивчення різних алгоритмів пошуку, таких як лінійний пошук, бінарний пошук, пошук з використанням хеш-таблиць тощо.
- Розбір принципів роботи та порівняння ефективності різних методів пошуку.

3. Практичні завдання:

- Виконання завдань з сортування та пошуку на конкретних даних.
- Реалізація алгоритмів сортування та пошуку на мовах програмування або використання відповідних інструментів онлайн.

Обов’язково, при поясненні видів сортування та алгоритмів пошуку використовувати анімації, зображення аби допомогти здобувачам освіти уявити та систематизувати отриману інформацію. Саме тому для вивчення цієї теми прекрасно підійдуть блок-схеми, або анімовані слайди, що покажуть принципи.

Вивчення теми "Обробка рядків" у старших класах профільного рівня інформатики є важливим для розуміння роботи з текстовими даними у програмуванні. Ось деякі методики, які можуть бути використані для ефективного вивчення цієї теми:

1. Основи рядків:

- Розуміння основних концепцій та операцій з рядками: конкатенація, зрізи (slicing), індексація тощо.
- Пояснення різниці між рядками та іншими типами даних.

2. Функції обробки рядків:

- Вивчення методів та функцій, які використовуються для обробки рядків у програмуванні (наприклад, методи рядків у мовах програмування, такі як Python, Java, C++).
- Практичні вправи з використанням функцій для зміни рядків, пошуку підрядків, розділення рядків тощо.

3. Регулярні вирази:

- Вивчення основних понять та синтаксису регулярних виразів для пошуку та обробки текстових даних.
- Практичні завдання з використанням регулярних виразів для пошуку шаблонів у тексті.

4. Практичні завдання:

- Застосування отриманих знань до практичних завдань, наприклад, обробка текстових файлів, аналіз тексту, робота зі строками на прикладних завданнях.

Під час вивчення цієї теми корисно стимулювати учнівську активність через виконання практичних завдань, обговорення різних підходів до обробки рядків, а також заохочувати створення власних програм чи проектів, які використовують рядки у програмуванні.

Вивчення теми "Графи" у профільному курсі інформатики у старших класах є важливим, оскільки графи широко застосовуються у різних галузях, зокрема в інформатиці, транспорті, комунікаціях та інших сферах. Ось деякі методики, які можна використовувати для ефективного вивчення цієї теми:

1. Основи теорії графів:

- Ознайомлення з основними поняттями теорії графів: вершини, ребра, напрямленість, цикли, шляхи тощо.
- Пояснення різних типів графів (наприклад, орієнтовані та неорієнтовані, зважені графи).

2. Алгоритми на графах:

- Вивчення основних алгоритмів для графів, таких як DFS (пошук в глибину), BFS (пошук в ширину), алгоритми пошуку найкоротшого шляху (наприклад, алгоритм Дейкстри, алгоритм Флойда-Уоршелла).
- Практичні вправи з використанням цих алгоритмів для розв'язання конкретних завдань на графах.

3. Представлення графів:

- Навчання методам представлення графів у програмуванні (матриця суміжності, список суміжних вершин тощо).
- Вирішення завдань з перетворення представлення графів одного типу на інший.

Під час вивчення теми "Графи" важливо надавати учням можливість застосовувати отримані знання на практиці, стимулювати їх для спільної роботи над проектами та практичними завданнями, що сприяє глибшому розумінню матеріалу та розвитку аналітичного мислення.

Вивчення теми "Динамічне програмування і жадібні алгоритми" є важливою частиною профільного курсу інформатики у старших класах, оскільки ці підходи широко використовуються для оптимізації рішень у складних задачах. Ось деякі методики, які можна використовувати для ефективного вивчення цієї теми:

1. Основи динамічного програмування:

- Пояснення основних принципів динамічного програмування (збереження підзадач, рекурсивні відношення, мемоізація).

- Розбір класичних прикладів задач, які можна вирішувати за допомогою динамічного програмування.

2. Жадібні алгоритми:

- Ознайомлення з поняттям жадібних алгоритмів та їх особливостями.
- Порівняння жадібних алгоритмів з іншими методами оптимізації.

3. Практичні вправи та приклади:

- Вирішення практичних завдань за допомогою жадібних алгоритмів та динамічного програмування.
- Виконання завдань, що демонструють переваги та обмеження цих методів.

4. Аналіз складності алгоритмів:

- Обговорення ефективності та складності алгоритмів, порівняння їх швидкості та працездатності у різних сценаріях.

5. Використання прикладних задач:

- Вирішення реальних прикладних задач за допомогою жадібних алгоритмів та динамічного програмування (наприклад, задачі на графах, задачі оптимізації тощо).

Під час вивчення цієї теми важливо спонукати учнів до активної участі, вирішення практичних задач, обговорення прикладів та порівнянь алгоритмів. Також це може бути чудовою можливістю для проведення практичних вправ, групових проектів та дослідницької роботи, що сприяє глибшому розумінню матеріалу та розвитку креативного мислення.

Вивчення теми "Основи обчислювальної геометрії" у старших класах профільного рівня інформатики може бути цікавим та корисним. Ця тема дозволяє учням зрозуміти, як комп'ютери працюють з геометричними об'єктами та допомагають розв'язувати реальні задачі. Ось деякі методики, які можна використовувати для вивчення цієї теми:

1. Основи геометричних об'єктів:

- Ознайомлення з базовими геометричними об'єктами, такими як точки, відрізки, вектори, прямі, площини та їх репрезентація у програмуванні.

2. Геометричні операції:

- Вивчення геометричних операцій, таких як обчислення відстаней між точками, знаходження перетину прямих, обчислення площі та периметру фігур тощо.

3. Алгоритми обчислювальної геометрії:

- Вивчення алгоритмів для вирішення конкретних задач, таких як перевірка належності точки до фігури, знаходження найближчої пари точок тощо.

4. Візуалізація та віртуальні інструменти:

- Використання спеціалізованих програм або бібліотек для візуалізації геометричних об'єктів та їх операцій.

5. Практичні завдання:

- Розв'язання практичних задач, які вимагають використання знань з обчислювальної геометрії, таких як задачі на побудову геометричних фігур, обчислення площі та об'єму, розв'язання задач відповідно до геометричних законів тощо.

Під час дистанційного навчання, можна використовувати різноманітні методи та ресурси для підтримки вивчення, такі як:

- Відеоуроки з поясненнями основних концепцій та прикладами використання алгоритмів на числах.
- Інтерактивні завдання та вправи для вирішення конкретних задач.
- Використання онлайн-інструментів для програмування та відлагодження коду.
- Віртуальні проекти або завдання, що стосуються реального світу для застосування вивчених концепцій.

Також важливо стимулювати учнівську активність через обговорення та спільну роботу над завданнями, що сприяє кращому засвоєнню матеріалу та розвитку аналітичного мислення.

2.2. Створення занять для вивчення розділу «Алгоритми» під час дистанційного навчання.

Для конкретних уроків було використано середовище для створення дизайну різного формату – Canva як основу, для створення візуальних об’єктів. Для демонтрації синтаксису, використання змінних тощо було використано інтерпретатор PyCharm, що має безкоштовну версію, або, як альтернатива застосунку, використано Online Python Compiler [11]– онлайн платформу, що дозволяє без встановлення та завантажування програми працювати з кодом. Оскільки уроки мають бути проведені у форматі дистанційного навчання – застосовується засіб Google Meet.

Урок на тему «Повторення вивчених структур даних. Прості змінні та масиви/списки» відноситься до модулю «Алгоритми» теми «Алгоритми сортування і пошуку даних» і є першим уроком даної теми. На уроці важливо повторити, освіжити знання здобувачів, які вони набули у попередніх класах стосовно структур даних, змінних та інших важливих елементах у програмуванні. Обраної мовою програмування є Python. Тому важливо ознайомити здобувачів з базовими знаннями, аби не спровокувати страх перед важким та невідомим.

Мета уроку: повторити та узагальнити знання з використання структур даних мовою Python: список, словник, кортеж, повторити знання про сталі, змінні, типи даних, структури даних.

Після завершення уроку здобувач освіти:

- знає:
    - комп'ютерну програму та її структуру мовою Python;
    - вказівки й алгоритми роботи у середовищі програмування;
    - поняття величини та її типу;
- вміє:

- застосовувати різні типи даних для різних варіантів завдань;
- оформлювати блочні структури;
- бути економним щодо кількості операцій і використання пам’яті;

Будова уроку:

1. Організація, привітання, перекличка.
2. Озвучування теми уроку та мети.
3. Теоретичний матеріал.
4. Повторення матеріалу шляхом розв’язування інтерактивних завдань.
5. Розв’язання завдань у групі.
6. Підведення підсумків, рефлексія.
7. Домашнє завдання.

Хід уроку:

1. Організація роботи.

Привітання, перевірка присутності озвучування потрібних матеріалів – ПК.

2. Озвучування теми уроку та мети.

Тема уроку: «Повторення вивчених структур даних. Прості змінні та масиви/списки»

Наша мета – повторити та закріпити знання з знання з використання структур даних мовою Python: список, словник, кортеж, повторити знання про сталі, змінні, типи даних, структури даних.

Чи пам’ятає хтось з минулих років, що таке структури даних? Чим вони характерні?

3. Теоретичний матеріал розміщений на слайді (рис. 2.1) (всі слайди презентації до уроку розміщені у Додаток А):

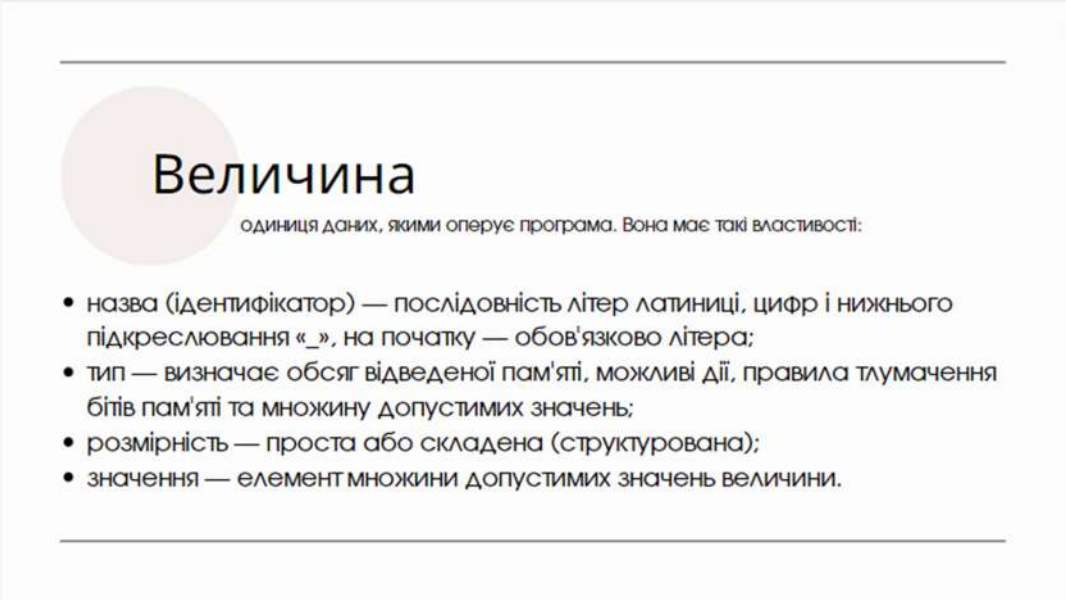


Рисунок 2.1 Слайд презентації до уроку

Кожній величині (або складовій для структурованої величини) в програмі відповідає певна область оперативної пам'яті, де зберігається це значення. Ця область має свою адресу, яка вказує на початок послідовності байтів, призначених для зберігання цієї величини. Назва змінної визначає ім'я цієї області пам'яті під час роботи програми.

У мові програмування Python кожне значення має свій тип, але тип змінної не потрібно явно оголошувати. Інтерпретатор Python самостійно визначає тип значення при присвоєнні і запам'ятовує його. Це означає, що у Python використовується динамічна типізація об'єктів: тип об'єкта можна змінити під час виконання коду, наприклад, при присвоєнні нового значення іншого типу.

4. Повторення матеріалу шляхом розв’язування інтерактивних завдань(рис.2.2).

Рисунок 2.2 Інтерактивне завдання до уроку на платформі LearningApps

До уроку розроблені кілька тестових та інтерактивних завдання, що допоможуть учням згадати потрібний матеріал.

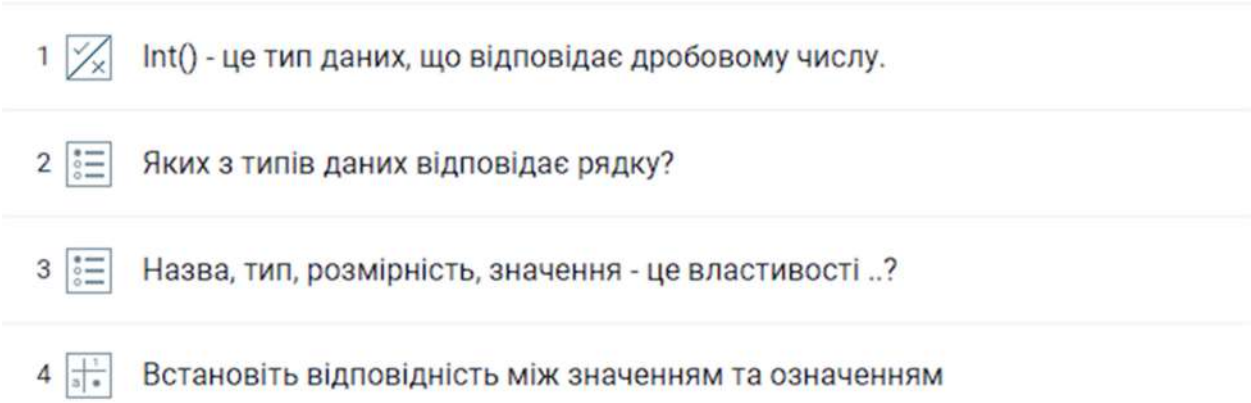


Рисунок 2.3 Тестові завдання різних типів на платформі ClassTime

Одним з завдань уроку є відкрити інтерпретатор та реалізувати наступні пункти:

- Створити два довільних списки.
- Видалити з першого списку другий елемент.
- Вивести змінений список на екран.
- Змінити у другому списку останній об'єкт.
- Вивести змінений список на екран.
- Створити новий список злиттям наявних.

- Вивести отриманий список на екран.

```
first_list = ['one', 'two', 'three', 'four', 'five']
second_list = [1, 2, 3, 4, 5]

first_list.pop(1)
print(first_list)

second_list[len(second_list)-1] = 6
print(second_list)

third_list = first_list + second_list
print(third_list)
```

Рисунок 2.4 Вирішення завдання.

5. Розв’язання задач у групі.
   - Зчитати 6 цілих чисел. Розмістити парні числа на початку списку, непарні — в кінці.
   - Заповнити випадковими числами різного знаку список з 20 елементів. Отримати новий список з додатних елементів списку з непарними індексами.
   - Заповнити випадковими невід'ємними цілими числами від 0 до 5 включно список з 20 елементів. Видалити зі створеного списку списку елементи, що належать проміжку [2, 4].
6. Підведення підсумків уроку.
7. Домашнє завдання: повторити та узагальнити матеріал створивши відповідні таблиці.

Урок на тему « Базові поняття обчислювальної геометрії»

Мета: вивчити, узагальнити та систематизувати знання про базові поняття обчислювальної геометрії. Вивчити поняття точка, відрізок, вектор, пряма, площина, координатна система, поверхня. Розглянути обчислювальні операції у геометрії та застосування обчислювальної геометрії у житті.

Будова уроку:

1. Організація, привітання, перекличка.
2. Озвучування теми уроку та мети.
3. Теоретичний матеріал.
4. Розв’язання завдань у групі.
5. Підведення підсумків, рефлексія.
6. Домашнє завдання.

Хід уроку:

1. Привітання, організаційні моменти.
2. Тема нашого уроку «Базові поняття обчислювальної геометрії» (Слайд 1). Пригадайте, які прості геометричні фігури ви використовували на уроках геометрії. (слайд 2)
3. Обчислювальна геометрія є важливою складовою сучасних технологій та має широкі застосування в різних сферах нашого життя. Ця галузь займається вивченням різноманітних геометричних об'єктів, таких як точки, відрізки, вектори, прямі, площини та поверхні, розробляючи при цьому алгоритми для вирішення геометричних завдань.(слайд 3)

   Основні поняття обчислювальної геометрі та їх характеристика (рис. 2.5):

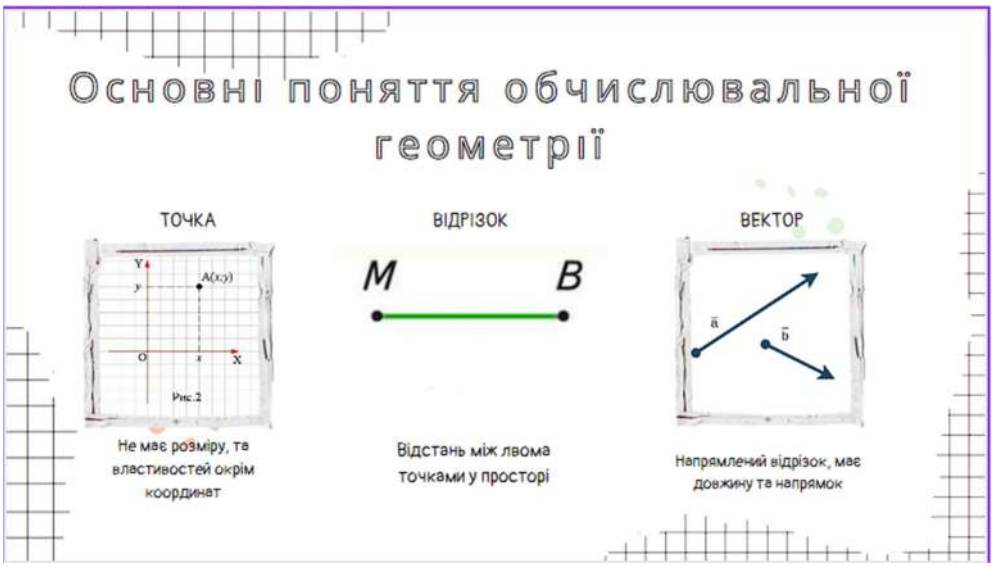


Рисунок 2.5 Слайд до теми "Базові поняття обчислювальної геометрії"

4. Завдання розроблені у Kahoot! заохотять здобувачів до навчального процесу (рис. 2.6):

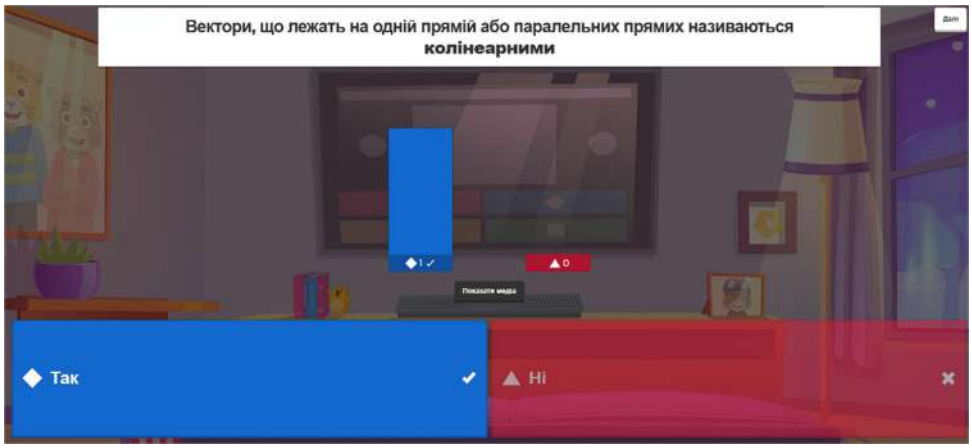


Рисунок 2.6 Вікторина до уроку

5. Підведення підсумків, рефлексія.

6. Домашнє завдання.

ВИСНОВКИ

У процесі дослідження було розібрано поняття дистанційного навчання та його історію становлення в Україні, структура модулю «Алгоритми» та основні поняття, що вони вміщають. Стало зрозуміло, що важливо створити можливості для практичного застосування знань учнів через вирішення різних задач та завдань, які вимагають використання обчислювальної геометрії. Також стимулювати учнівську активність через спільну роботу над проектами та практичними завданнями, що дозволяє краще засвоєння матеріалу та розвиток творчого мислення.

У результаті роботи над кваліфікаційної роботою було досягнуто поставлену мету – дослідити структуру розділу «Алгоритми» та методичні аспекти його вивчення під час дистанційного навчання.

Для досягнення даної мети були виконані всі поставлені завдання:

1. Досліджено структуру та вміст розділу «Алгоритми» інформатики профільного рівня для старших класів.
2. Досліджено поняття дистанційного навчання та історію його становлення в Україні.
3. Розглянуто методичні аспекти вивчення розділу «Алгоритми» під час дистанційного навчання, методи та способи урізноманітнення навчання.
4. Розроблено уроки для вивчення розділу «Алгоритми» під час дистанційного навчання з використанням інтерактивних завдань.

Окрім цього, у першому розділі кваліфікаційної роботи була розібрана структура розділу «Алгоритми», його особливості та основні терміни. Було, також, визначено передумови для вивчення даного розділу (що мали вивчати здобувачі освіти до даного розділу, аби добре засвоїти знання) та визначено якими мають бути результати навчання (якими знаннями, вміннями та цінностями мають володіти здобувачі освіти після вивчення курсу). Виокремлено мову програмування (python), яку доцільно використовувати у вивченні парадигм та технологій програмування, порівнявши її з іншою мовою, що також є досить актуальною у сучасному світі (C++). У результаті

даного вибору мови програмування було проаналізовано різні середовища програмування, виокремлені їх функції та особливості. Описані середовища програмування (Python IDLE, PyCharm, PyDev, Komodo IDE, Spyder) кожен мають свої особливості та переваги, тому у виборі між середовищами програмування можна обрати до вподоби. У процесі дослідження та опрацювання джерел, стає зрозуміло, що найкращим вибором серед мов програмування є python, через велику частку переваг. Проте вибір серед середовищ програмування кожен заклад освіти/педагог робить самостійно, відштовхуючись від вимог, особливостей та функцій.

У другому розділі досліджено методику вивчення модулю «Алгоритми», запропоновано тематичне планування, яке доцільно використовувати при плануванні вивчення розділу, розроблено уроки на теми « Повторення вивчених структур даних. Прості змінні та масиви/списки» та «Базові поняття обчислювальної геометрії».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. An Empirical Study of Novice Program Comprehension in the Imperative and Object-Oriented Styles. URL: https://dl.acm.org/doi/pdf/10.1145/266399.266411 (дата звернення: 01.10.2022).

2. BANAS, Edward J.; EMORY, W. Frances. History and issues of distance learning. Public administration quarterly, 1998, 365-383.

3. Downey A, Elkner J and Meyers C 2008 "`How to think like Computer Scientist` Learning with Python"

4. Empirical comparison of objects-first and objects-later. URL: https://dl.acm.org/doi/pdf/10.1145/1584322.1584326?casa_token=xL2ji8lMSTUAAAAA:zivCJixx7uP2Q3K3ECycHzMuGBZpjNicBpHBhtvMSrTB4YsK0H-0B607TzZbMp1_SyriZWDwYp-D (дата звернення: 01.10.2022).

5. Features – PyCharm: веб-сайт. URL: https://www.jetbrains.com/pycharm/features/ (дата звернення 14.05.2023)

6. Ilomäki, Liisa, Anna Kantosalo, and Minna Lakkala. "What is digital competence?" *Linked portal*, веб-сайт. URL: https://helda.helsinki.fi/bitstream/handle/10138/154423/Ilom_ki_etal_2011_What_is_digital_competence.pdf?sequence=1 (дата звернення 29.10.2022).

7. Journal of Physics: Conference Series, Volume 423, 2013 International Conference on Science & Engineering in Mathematics, Chemistry and Physics (ScieTech 2013) 24–25 January 2013, Jakarta, Indonesia

8. KING, Frederick B., et al. Defining distance learning and distance education. AACE Review (Formerly AACE Journal), 2001, 9.1: 1-14.

9. Komodo IDE By ActiveState - One IDE for All Your Languages: веб-сайт. URL: https://www.activestate.com/products/komodo-ide/ (дата звернення 05.05.2023)

10. ONG, Shyue Ping, et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. Computational Materials Science, 2013, 68: 314-319.

11. Programiz. Python Online Compiler, веб-сайт. URL: https://www.programiz.com/python-programming/online-compiler/ (дата звернення 29.10.2022).

12. PyDev: веб-сайт.URL: https://www.pydev.org/ (дата звернення 14.05.2023)

13. Python Tutor: Learn Python, JavaScript, C, C++, and Java programming by visualizing code: веб-сайт. URL: https://pythontutor.com/ (дата звернення 05.05.2023)

14. Sanders ID and Langford S ”Students' Perceptions of Python as a First Programming Language at Wits”

15. Spyder IDE: веб сайт. URL: https://www.spyder-ide.org/ (дата звернення 05.05.2023)

16. The Effects of Objects-First and Objects-Late Methods on Achievements of OOP Learners. URL: https://www.scirp.org/html/23962.html?pagespeed=noscript (дата звернення: 01.10.2022).

17. VAN ROSSUM, Guido; DRAKE, Fred L. Python library reference. 1995.

18. Базові поняття обчислювальної геометрії. Canva, веб-сайт. URL: https://www.canva.com/design/DAF2hMHQ-e0/64SfMEAc2dQU8bk3uJX-kA/view?utm_content=DAF2hMHQ-e0&utm_campaign=designshare&utm_medium=link&utm_source=editor (дата звернення 29.10.2022).

19. Базурін, В., Омелечко, Є., & Ковтун, А. (2018). Comparative analysis of development environments on Python language. New Computer Technology, 16, 281-292.

20. IDLE – Python 3.11.3 documentation: веб-сайт. URL: https://docs.python.org/3/library/idle.html (дата звернення 14.05.2023)

21. Календарно-тематичне планування уроків інформатики (профільний рівень) для 11 класу, веб-сайт. URL: https://teach-inf.com.ua/load/kalendarni_plani/10_11_klas/kalendarno_tematichne_planuvannja_informatika_11_klas_profilnij_riven_175_god_rik_5_god_tizhden_2019/173-1-0-2505 (дата звернення 29.10.2022).

22. КУХАРЕНКО, В. М.; БОНДАРЕНКО, В. В. Екстрене дистанційне навчання в Україні. Харків: Міська друкарня, 2020, 7-29.

23. Махенько Я., Стельмашенко Я. Вивчення об'єктно-орієнтованого програмування в школі. IV Всеукр. наук.-практ. онл.-фор. «Іноваційні трансформації в сучасній освіті: виклики, реалії, стратегії», 27 жовт. 2022 р. Київ: ДНУ, 2022.

24. Махенько Я., Стельмашенко Я. Структура шкільного курсу інформатики в старших класах. VII Всеукр. наук.-практ. конф. з міжн. уч. «Сучасні інформаційні технології в освіті та науці»,17-18 лист. 2022 р. Житомир: ЖДУ, 2022.

25. Навчальна програма профільного рівня для 10-11 класів з інформатики, веб-сайт. URL: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv (дата звернення: 01.11.2022).

26. Навчальна програма рівня стандарту для 10-11 класів з інформатики, веб-сайт. URL: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv (дата звернення: 01.11.2022).

27. ОСАДЧА, К. П.; ХРОМИШЕВ, О. В. Аналіз методів розробки алгоритмів розв'язання математичних задач засобами мови Python. Системи обробки інформації, 2016, 2: 114-117.

28. Повторення вивчених структур даних. Прості змінні та масиви/списки. Canva, веб-сайт. URL:

https://www.canva.com/design/DAF2fp7jlkw/mumkEHPSy3dMEbD25MAQSw/view?utm_content=DAF2fp7jlkw&utm_campaign=designshare&utm_medium=link&utm_source=editor (дата звернення 29.10.2022).

29. ПРИБИЛОВА, Вікторія Миколаївна. Проблеми та переваги дистанційного навчання у вищих навчальних закладах України. Проблеми сучасної освіти, 2013, 4.

30. Руденко В. Інформатика (профільний рівень): підручний 11 кл. закл. заг. серед. освіти. Харків, вид-во «Ранок», 2019.

31. ХАТУНЦЕВ, Андрій Юрійович; МАРТИНОВА, Наталія Сергіївна. Обчислювальна геометрія та комп'ютерна графіка. 2009.

32. ЧОРНА, А. В.; ТАГАНОВА, Д. В. Використання алгоритмів розв'язування графічних задач засобами Python. Інформаційні технології в освіті та науці: зб. наук. пр., 2018, 10: 310-315.

33. PHILLIPS, Dusty. Python 3 object oriented programming. Packt Publishing Ltd, 2010

34. The Effects of Objects-First and Objects-Late Methods on Achievements of OOP Learners. URL: https://www.scirp.org/html/23962.html?pagespeed=noscript (дата звернення: 01.10.2022).

35. Empirical comparison of objects-first and objects-later. URL: https://dl.acm.org/doi/pdf/10.1145/1584322.1584326?casa_token=xL2ji8lMSTUAAAAA:zivCJixx7uP2Q3K3ECycHzMuGBZpjNicBpHBhtvMSrTB4YsK0H-0B607TzZbMp1_SyriZWDwYp-D (дата звернення: 01.10.2022).

36. An Empirical Study of Novice Program Comprehension in the Imperative and Object-Oriented Styles. URL: https://dl.acm.org/doi/pdf/10.1145/266399.266411 (дата звернення: 01.10.2022).

37. Конспект лекції No 8-9-10Тема No 5. КОНЦЕПЦІЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ URL: https://financial.lnu.edu.ua/wp-

content/uploads/2017/09/lektsiia_OOP_8_9_10_tema-5.pdf (дата звернення: 03.10.2022).

38. J. Gosling, B. Joy, G. Steele, G. Brachda. The Java Language Specification, 2nd Edition URL: https://docs.oracle.com/javase/specs/jls/se19/jls19.pdf (дата звернення: 03.10.2022).

39. Microsoft. The modern, innovative, open-source programming language for building all your apps. URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата звернення: 04.10.2022).

10. Editors' Report -- Programming Languages -- C++ URL: https://web.archive.org/web/20200204081709/http://open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4661.html (дата звернення: 04.10.2022).

41. Офіційна сторінка Python URL: https://www.python.org/ (дата звернення: 03.10.2022).

42. Python 3.10.1 is available URL: https://web.archive.org/web/20220106181415/https://blog.python.org/2021/12/python-3101-is-available.html (дата звернення: 02.10.2022).

43. The Python tutorial, URL: https://docs.python.org/3/tutorial/ (дата звернення: 01.10.2022).

44. Django, URL: https://www.djangoproject.com/ (дата звернення: 05.10.2022).

АНОТАЦІЯ

У кваліфікаційній роботі студентки 25Мд-СО(інф) групи Стельмашенко Яніни на тему «Методичні аспекти вивчення розділу «Алгоритми та програми» під час дистанційного навчання» метою стало дослідити структуру розділу «Алгоритми» та методичні аспекти його вивчення під час дистанційного навчання. Об'єктом дослідження є методика вивчення розділу «Алгоритми» шкільного курсу інформатики, а предметом – елементи методичної системи, що сприяють опануванню учнями розділу «Алгоритми» шкільного курсу інформатики. У процесі дослідження були використані такі методи дослідження, як: порівняльний, експериментальний, структурний, описовий.

У кваліфікаційні роботі розглядається поняття дистанційного навчання та його розвиток у історії України. Досліджена структура та зміст модулю «Алгоритми», його основні поняття. Аналізуються мови програмування, які доцільно використовувати у процесі вивчення модулю «Алгоритми». Описується методика вивчення модулю під час дистанційного навчання.

Результатом проведеного дослідження стало зрозуміло, що важливо створити можливості для практичного застосування знань учнів через вирішення різних задач та завдань, які вимагають використання обчислювальної геометрії. Також стимулювати учнівську активність через спільну роботу над проектами та практичними завданнями, що дозволяє краще засвоєння матеріалу та розвиток творчого мислення. У результаті роботи над кваліфікаційною роботою створено уроки на теми « Повторення вивчених структур даних. Прості змінні та масиви/списки» та «Базові поняття обчислювальної геометрії».

Ключові слова: інформатика, дистанційне навчання, Python, мова програмування, алгоритми, сортування, типи даних, методика навчання інформатики.

ABSTRACT

In the qualification work of the student of the 25Md-SO (inf) group, Yanina Stelmashenko, entitled "Methodological Aspects of Studying the Section 'Algorithms and Programs' During Distance Learning," the aim was to investigate the structure of the "Algorithms" section and the methodological aspects of its study during distance learning. The object of the research is the methodology of studying the "Algorithms" section of the school computer science course, and the subject is the elements of the methodological system that contribute to students' mastery of the "Algorithms" section of the school computer science course. The research utilized such research methods as comparative, experimental, structural, and descriptive.

The qualification work examines the concept of distance learning and its development in the history of Ukraine. It explores the structure and content of the "Algorithms" module, including its fundamental concepts. Programming languages suitable for studying the "Algorithms" module are analyzed. Additionally, the methodology of teaching the module during distance learning is described.

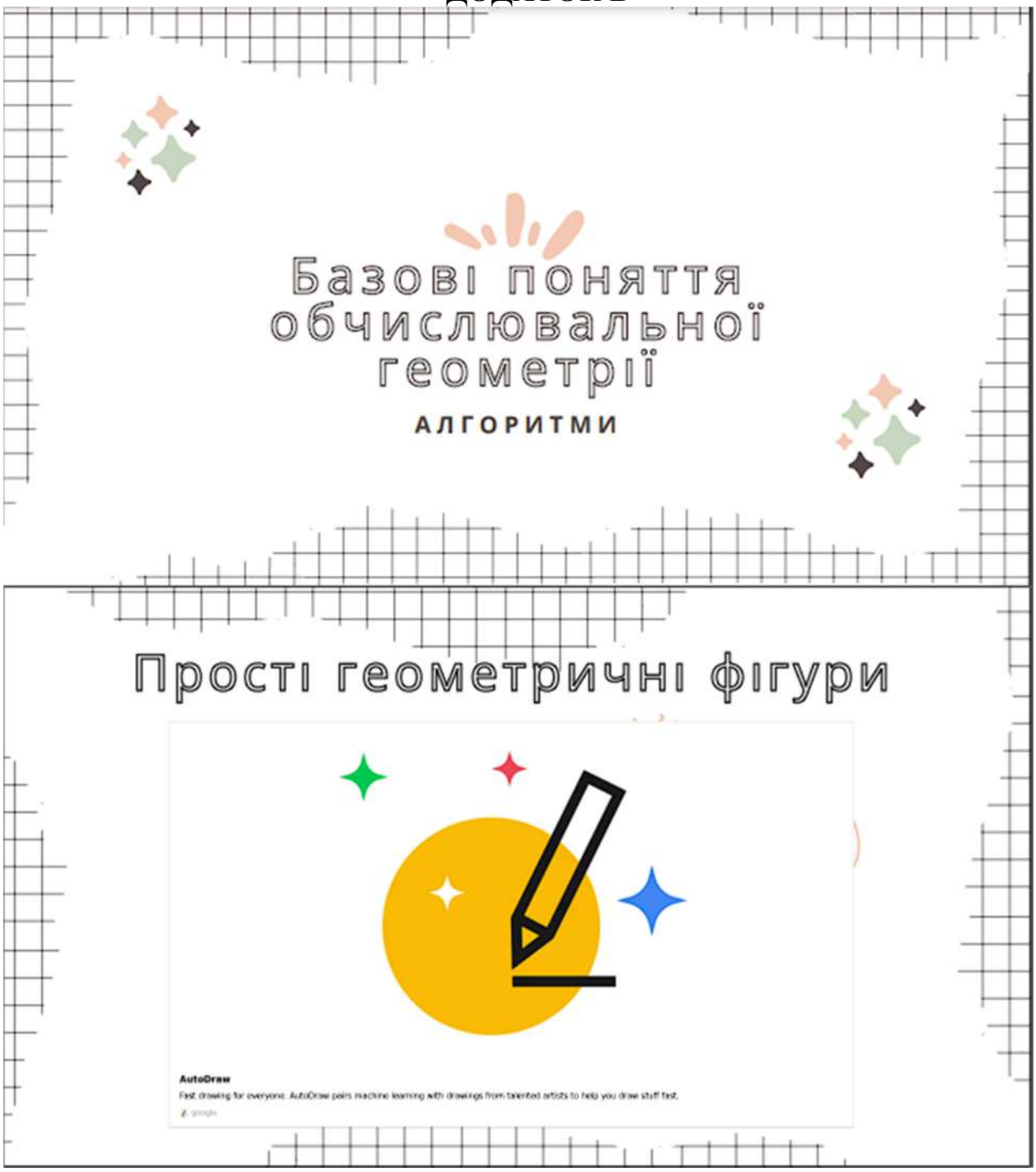
The result of the conducted research reveals the importance of creating opportunities for the practical application of students' knowledge through solving various tasks that require the use of computational geometry. It also emphasizes stimulating students' activity through collaborative work on projects and practical tasks, which enhances better assimilation of the material and fosters creative thinking. As a result of working on the qualification work, lessons on the topics of "Review of Learned Data Structures. Simple Variables and Arrays/Lists" and "Basic Concepts of Computational Geometry" have been created.

Keywords: computer science, distance learning, Python, programming language, algorithms, sorting, data types, computer science teaching methodology.

ДОДАТОК А


Найпопулярніші типи даних Python
TRUE АБО FALSE
ЧИСЛО
РЯДОК
СПИСОК
Логічний
Ціле, з крапкою, дріб, комплекне число
Рядок тексту
послідовність значень
>>> type(a)
<Class 'int'>
>>> type(81)
<Class 'int'>
>>> type(99.99)
<Class 'float'>
>>> type('ruby')
<Class 'str'>

ДОДАТОК Б


Базові поняття обчислювальної геометрії
АЛГОРИТМИ
Прості геометричні фігури
AutoDraw
Fast drawing for everyone. AutoDraw pairs machine learning with drawings from talented artists to help you draw stuff fast.

Основні поняття обчислювальної геометрії

ТОЧКА

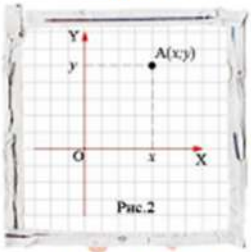


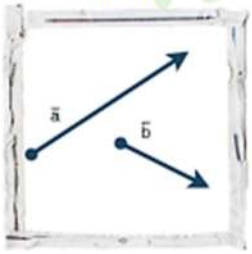
Рис.2

Не має розміру, та властивостей окрім координат

ВІДРІЗОК

Відстань між двома точками у просторі

ВЕКТОР



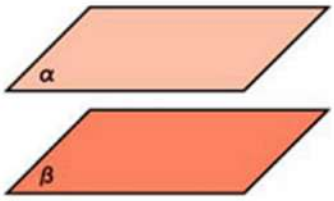
Напрямлений відрізок, має довжину та напрямок

Основні поняття обчислювальної геометрії

ПРЯМА

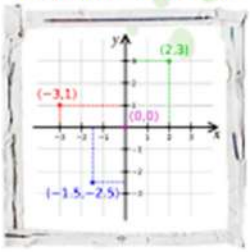
набір точок, які лежать на одній лінії та не мають початку чи кінця

Площина



рівнина, яка складається з усіх точок у тривимірному просторі

Координатна система



система, яка використовується для визначення положення точок у просторі.