ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ

ФОРМА НАВЧАННЯ ДЕННА

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець

(підпис)

«_____»_____2020р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему
СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ТЕМОЮ «АЛГОРИТМИ СОРТУВАННЯ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Писаренко Ігор Андрійович

_____ «___»_____2020р.

Науковий керівникк.ф.-м.н., доц.,Чілікіна Тетяна Василівна

_____ «___»_____2020р.

Полтава 2020

Вищий навчальний заклад Укоопспілки

«Полтавський університет економіки і торгівлі»

ЗАТВЕРДЖУЮ

Зав. кафедри ММСІ

_____ О.О. Ємець

(підпис)

«__» _____2019 р.

Завдання та календарний графік

виконання бакалаврської роботи

Студента спеціальності 122 «Комп'ютерні науки»

Прізвище , імя, по батькові Писаренко Ігор Андрійович

1. Тема роботи Створення програмного забезпечення за темою «Алгоритми сортування»

затверджена наказом ректора №151-Н від «02»вересня 2019 р.

Термін подання студентом дипломної роботи «20» травня 2020 р.

2.Вихідні дані до дипломної роботи

Публікації по темі роботи, методичні рекомендації, стандарти.

3. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП

1. ПОСТАНОВКА ЗАДАЧІ

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Необхідність та актуальність данної теми

2.2. Огляд деякого програмного забезпечення для дистанційного навчання

2.3. Програми – тренажери в дистанційній освіті

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритмізація тренажера
3.2. Розробка блок-схеми, яка підлягають програмуванню
3.2. Алгоритми сортування
4. ПРАКТИЧНА ЧАСТИНА
4.1. Опис процесу програмної реалізації
4.2 Технології реалізації тренажеру
4.3. Технології реалізації тренажеру
4.4. Опис програми.
4.5. Інструкція користувача.
ВИСНОВКИ
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ
ДОДАТОК А

4. Перелік графічного матеріалу (з точним визначенням кількості блок-схем, іншого графічного матеріалу)

Блок-схема алгоритму

5. Консультанти розділів дипломної роботи

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
Вступ	ЧілікінаТ.В.	30.09.19	
1. Постанова задачі	ЧілікінаТ.В.	30.09.19	
2. Інформаційний огляд	ЧілікінаТ.В.	30.09.19	
3. Теоретична частина	ЧілікінаТ.В.	30.09.19	
4.Практична реалізація	ЧілікінаТ.В.	30.09.19	

6. Календарний графік виконання дипломної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ	10.05.20	
2. Вивченняметодичнихрекомендацій та стандартів та звіткерівнику	01.03.20	
3. Постановка задачі	01.03.20	
4. Інформаційнийоглядджерелбібліотек та інтернету	15.03.20	
5. Теоретична частина	15.04.20	
6. Практична частина	01.05.20	
7. Закінченняоформлення	15.05.20	
8. Доповідь студента накафедрі	15.05.20	
9. Доробка (за необхідністю), рецензування	15.06.20	

Дата видачі завдання «___» _____ 201_р.

Студент(ка) _____

(підпис)

Науковий керівник _____ _____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту дипломної роботи

Дипломна робота оцінена на _____

(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 201_ р.

Секретар ЕК_____ _____

(підпис) (ініціали та прізвище)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 6
ВСТУП 8
1 ПОСТАНОВКА ЗАДАЧІ 10
2 ІНФОРМАЦІЙНИЙ ОГЛЯД 12
2.1.Необхідність та актуальність данної теми 12
2.2 Огляд деякого програмного забезпечення для дистанційного навчання 18
2.3 Програми – тренажери в дистанційній освіті 22
3 ТЕОРЕТИЧНА ЧАСТИНА 23
3.1 Алгоритмізація задачі за темою роботи. 23
3.2 Розробка блок-схеми, яка підлягають програмуванню. 25
3.3 Алгоритми сортування. 26
4 ПРАКТИЧНА ЧАСТИНА 29
4.1 Опис процесу програмної реалізації 29
4.2 Технології реалізації тренажеру 32
4.3Опис програми. 34
4.4 Інструкція користувача 43
ВИСНОВКИ 46
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 47
ДОДАТОК А 48

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
ДК	Дистанційний курс
JSON	Java script object notation - формат збереження даних

РЕФЕРАТ

Записка: 76 сторінок, 15 рисунків, 1 додатка (на 29 сторінках), 8 літературних джерел.

Предмет розробки – є програмний продукт, який реалізує тему «Алгоритми сортування» на мові програмування JavaScript.

Мета роботи – створити програмне забезпечення з теми «Алгоритми сортування».

Методами розробки є використання редактора Visual Studio Code на мові програмування JavaScript та мовах розмітки HTML5, CSS3.

Ключові слова: ТРЕНАЖЕР, WEB-ДОДАТОК, JS, HTML5, CSS3

ВСТУП

У сфері освіти зростає роль та використання інформаційних технологій. Головним трендом у сфері освіти стають відкриті та закриті онлайн-курси і освіта основана на медіа технологіях. Курси що проводяться в онлайн режимі та тести стали сьогодні дуже популярним засобом для навчання, така форма навчання дає змогу інтерактивного спілкування між студентами та викладачами, а також онлайн прийому іспитів, наприклад це використовується для отримання цифрових сертифікатів в різних компаніях таких як Microsoft. Це одна із найсучасніших форм дистанційного навчання, що активно розвивається у освіті всього світу.

Теоретичні знання отримані студентом під час роботи з дистанційними матеріалами потребують перевірки за допомогою спеціальних тренажерів і тестових засобів. Вони дозволяють не тільки перевірити знання і навички, а також можуть навчати в інтерактивній формі. Наприклад тренажер в якому студент може самостійно вводити дані та перевіряти результат, і якщо помилився отримувати підказки та виправлення. Така інтерактивна форма допомога краще засвоїти знання, на відміну від простого читання інформації по темі.

Актуальність проекту полягає в створенні програмного забезпечення, що дає можливість студенту застосувати його для самостійного та дистанційного вивчення «Алгоритмів сортування». Це в подальшому дозволить легко використовувати дану циклічну структуру, що спрощує час в написані коду та автоматизує процес.

Мета бакалаврської работи – створити програмне забезпечення з теми «Алгоритми сортування»

Об'єктом розробки- створення програмного забезпечення у вигляді тренажеру з теми «Алгоритми сортування».

Предметом розробки є програмний продукт, який реалізує тему «Алгоритми сортування» на мові програмування JavaScript.

Методи розробки –методи алгоритмів сортування, об’єктно-орієнтовану мову програмування JavaScript та технології та інструментальних засоби створення комп’ютерних та web додатків.

Бакалаврська робота складається з трьох розділів, а саме: постановка задачі, інформаційний огляд, теоретична частина, практична частина.

Результатом виконання бакалаврської роботи є програмне забезпечення з теми «Алгоритми сортування» на мові програмування JavaScript.

Обсяг пояснювальної записки: _76_ стор., в т.ч. основна частина _43_ стор., додатки - _29_ стор., джерел - _8_ назв.

1 ПОСТАНОВКА ЗАДАЧІ

В бакалаврській роботі потрібно розробити програмне забезпечення у вигляді тренажер з теми «Алгоритми сортування», за допомогою якого студенти можуть перевіряти та закріпляти свої знання з даної теми. Тренажер містить як практичні так і теоретичні питання. Для проходження тестування тренажер повинен мати тести трьох типів. При проходженні тесту студент має відповісти на теоретичні запитання, якщо студент відповів правильно відбудеться перехід на сторінку з задачами по алгоритму, якщо студент відповідає не правильно – відбудеться перехід на сторінку с теоретичною довідкою, після ознайомлення з якою студент може знову спробувати відповісти на запитання або повернутися на сторінку з усіма алгоритмами.

Завдання в тестах мають бути двох типів складності:

1. теоретичне завдання з вибором варіанту відповіді;
2. практичне завдання з внесенням відповіді з клавіатури.

Проектування тренажеру складається з наступних кроків.

Крок 1. Формування вимог до тренажеру. Даний крок включає в себе розробку та узгодження з викладачем логічної структури тренажеру, а також дизайну за потребою. Розробка алгоритму його роботи.

Крок 2. Вибір середовища та мови програмування. Цей крок є важливим, так як він вирішує багато питань пов’язаних як з можливістю роботи тренажера на різних системах.

Крок 3. Програмна реалізація на обраній мові програмування з урахуванням всіх вимог до тренажеру.

Крок 4. Тестування та вдосконалення розробленого програмного продукту. На цьому кроці потрібно повністю перевірити програму, її складові частини, роботу в середовищах. За потребою можливо вдосконалити тренажер, додавши нові можливості чи компоненти.

Дотримання наведених кроків і вимог до тренажеру, забезпечує процес створення алгоритму і програмної реалізації всіх елементів програмного забезпечення.

2 ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Необхідність та актуальність данної теми

Найважливішим компонентом педагогічної системи і складовою частиною навчального процесу є контроль за навчальною діяльністю учнів, призначений для визначення успішності навчання кожного учня, аналізу отриманих результатів та корекції подальшого процесу навчання. Інструментом підвищення якості освіти разом із реформуванням змісту освіти виступає вдосконалення системи оцінювання, модернізація діагностики навчання учнів.

Останнім часом замість традиційного поняття "контроль", окрім вже згадуваного поняття "діагностика", все частіше стали використовувати поняття "моніторинг". При проектуванні навчального процесу перед педагогом постає завдання вибору методів і форм контролю навчальних досягнень учнів, критеріїв якості засвоєння вивченого матеріалу, розробки процедур здійснення контролю, обґрунтування способів індивідуальної корекції навчальної діяльності учнів.

Необхідність забезпечення контролю й оцінювання не тільки результату, а й процесу навчання сприяє пошуку оперативних та об'єктивних методів контролю знань. Система оцінки і контролю повинна відповідати вимогам управління пізнавальною діяльністю учнів і виступати в ролі відповідного інструментарію для її здійснення.

Суб'єктивність оцінки знань пов'язана певною мірою з недостатньою розробкою методів контролю системи знань. Об'єктивний підхід полягає в тому, що для виявлення наявності знань завжди використовується адекватний інструмент. Сучасна методика пропонує тест як інструмент вимірювання рівня знань, за допомогою якого можна не тільки виявити якість навчання, але і оптимально управляти навчальним процесом.

Ефективність функціонування системи професійної освіти значною мірою залежить від впровадження в діяльність вузу інновацій. Вони змушують всіх учасників навчально-виховного процесу визначити й проаналізувати рівень своїх

знань, умінь, навичок, спрямувати свою діяльність на шлях перспективних перетворень. При цьому основна роль належить викладачеві. Його готовність до впровадження інновацій, вміння організувати цей процес та управляти ним є невід'ємними складовими успішної інноваційної роботи.

Однією з інноваційних форм інформатизованої системи освіти є тестування. Ця форма, надзвичайно популярна за рубежем, в останні десятиліття активно впроваджується і в Україні. Національний університет "Києво-Могилянська Академія" був першим вищим навчальним закладом нашої держави, який під час відбору студентів почав застосовувати систему тестування. Цю систему було визначено однією з кращих в Європі. Дану практику дедалі частіше впроваджують інші вищі навчальні заклади. Тому в підготовці молодшого спеціаліста використання тестування як методу контролю та оцінки знань є актуальним.

Тест (test) – слово англійського походження, що означає іспит, спробу, випробування. Тест – це стандартизоване завдання, за результатами якого роблять висновок про знання, уміння, навички (здібності, професійну придатність, обдарованість тощо) того, кого випробовують. У сучасній теорії та практиці тестового контролю нараховується понад 20 різновидів тестів: залежно від мети, характеру та функцій контролю, характеру, форми відповіді.

За результатами опитування 32% викладачів вищих навчальних закладів постійно використовують тести для контролю, 68% - користуються епізодично, або не користуватися ними зовсім. Близько 80% викладачів, які використовують тестовий контроль, працюють за авторськими тестами, при цьому вони орієнтуються на стандартні тестові методи, що як правило передбачають закриті тести з двома – трьома варіантами відповідей. Запитання, як правило, мають випадковий характер, відображаючи емпіричне узагальнення і формування вміння діяти за зразком. Тобто, недоліки традиційного розумово-емпіричного навчання розповсюджується і на зміст тестування.

Цілком очевидною є необхідність не тільки більш широкого та активного використання тестових методів у викладанні спеціальних, соціально-економічних та фундаментальних дисциплін, а й розробки методичних матеріалів.

Тест як система завдань специфічної форми і відповідного змісту є науково обґрунтованим інструментом оцінювання знань, умінь і навичок студентів, допомагає здійснювати індивідуальний контроль результатів навчання кожного з них, мобільно керувати навчально-виховним процесом. Порівняно з традиційними формами контролю знань (контрольна робота, іспит, залік, диференційованих залік.) тестування нерідко виявляється більш ефективно. Адже недостатньо оцінити рівень знань, треба спрогнозувати, як студент зможе ці знання використати. Тобто, зростає роль психологічного тестування.

Виникає необхідність поєднати його з тестуванням рівня знань. Тоді результати будуть більш об’єктивними, враховуватимуть не лише рівень та обсяг знань, а й особливості характеру студента, його нахили та можливості освіти, вміння аналізувати, узагальнювати, встановлювати причинно-наслідкові зв’язки. Як свідчить досвід, найчастіше оцінювання і контроль досягнень студента здійснюється за кінцевим результатом. На жаль, майже не піддається оцінці його діяльність, рівень розвитку в динаміці, вплив зовнішніх факторів на момент оцінювання.

Практика показала наступні переваги тестування над іншими формами контролю знань:

— упродовж досить обмеженого часу може бути перевірена якість знань, навичок у зазначеної кількості студентів;

— можливий контроль знань, умінь, навичок на необхідному, заздалегідь запланованому рівні;

— реальним є самоконтроль;

— знання оцінюють більш-менш об’єктивно;

— увага студента фіксується не на формуванні відповіді, а не осмисленні її суті;

— створюють умови для постійного зворотного зв’язку між студентом і викладачем.

— Проте тестовий контроль знань має й істотні недоліки:

— ймовірність випадкового вибору правильної відповіді;

— можливість при застосуванні тестів закритого типу оцінки тільки кінцевий результат (правильно - неправильно), у той час як сам процес, що привів до нього, не розкривається;

— психологічний недолік – стандартизація мислення без врахування рівня розвитку особистості;

— велика затрата часу на складання необхідного "банку" тестів, їх варіантів, трудомісткість процесу;

— тести не сприяють розвитку мови.

У пошуках шляхів підвищення ефективності системи перевірки рівня засвоєння студентами курсу "Діловодство" було введено систематичне тестування. Тести при вивченні діловодства впроваджувались поступово, що дало змогу психологічно підготувати студентів. Спочатку пропонувались прості тести з вибірковими відповідями і лише через деякий час вводились більш складні конструкції. Доведено, що надзвичайно продуктивним є використання навчальних тестів, підготовлених на основі методу ключових ситуацій.

Інформація в питанні тесту може подаватись у будь-якій формі – тексту, графічного зображення, звукового повідомлення, відео сюжету, формули тощо. Причому, ми ввели в практику відкритість та доступність бази тестів, тобто зміст тестів відомий студентам. Але при цьому тести постійно поновлюються з урахуванням змін у навчальних планах, програмах, реаліях сьогодення. Тестування з діловодства гарантує об’єктивність оцінки знань, умінь, навичок

студентів, сприяє усуненню проявів суб'єктивізму, а відтак і формуванню позитивного ставлення до даної дисципліни і викладача.

Дуже важливим при проведені тестового контролю є дотримання організаційного моменту (пояснення мети, порядку виконання та оформлення тесту, визначення часу та його виконання, забезпечення кожного студента бланком відповідей стандартного зразка, що великою мірою заощаджує час і студента, і викладача).

Тестові завдання значно скорочують час очікування студентами оцінки, що є суттєвим фактором – к психологічним, так і виховним.

Після тестування, а воно може охоплювати матеріал окремої теми, розділу або курсу в цілому, обов'язково аналізують його результати. Аналіз необхідний для того, щоб студент зміг перевірити, наскільки адекватно він оцінює свої знання, повірити у власні сили і скорегувати свою підготовку. Викладач не лише фіксує факт помилок і називає правильні відповіді, а й докладно пояснює студентам причини помилкових дій.

Студентам, які допустили помилки, пропонується письмово або усно відповісти на запитання: Чому вибрана відповідь неправильна? Наведіть докази на користь правильного варіанта. Робота над помилковими відповідями, усвідомлення помилки, її причини, знайдення помилкової ланки в ланцюгу розумових дій значно зменшують ймовірність запам'ятовування помилкових знань, "витіснення" правильних відповідей. Таким чином викорінюється практика "вгадування" студентом правильного рішення поставленої в тесті задачі.

Наступним етапом впровадження та використання тестування в процесі вивчення дисципліни "Діловодство" стала їх електронна форма. Було забезпечено їх функціонування на локальному комп'ютері та в локальній мережі навчальної аудиторії. Така форма, як правило, викликає додатковий інтерес у студентів, а крім того, дозволяє їм самостійно без участі викладача перевірити та оцінити рівень власних знань з конкретної теми чи комплексу тем курсу. У

комп’ютерній програмі тестування використовуються анімаційні, звукові, ігрові елементи, а також система реєстрації ходу перевірки та її результатів, зокрема, таких показників, як час розв’язання кожної задачі, кількість помилкових і правильних відповідей, підсумкова оцінка.

Важливою умовою тестування, як універсального педагогічного інструмента, є частота його проведення, яка залежить від дисципліни, її ролі і місця в навчальному плані, особливостей засвоєння знань. Слід зробити тестування звичною і зручною формою регулярного контролю знань студентів. Необхідно пам’ятати, що тестування – це не самоціль, а ефективна форма повторення – узагальнення і впорядкування вивченого. Контрольно-оцінювальна функція навчання – це лише елемент добре організованого і технологічно продуманого навчально-виховного процесу. Якщо студенти матимуть міцні знання, то їх оцінювання не становитиме особливих труднощів, в якій би формі воно не проводилось.

2.2. Огляд деякого програмного забезпечення для дистанційного навчання

Однією з важливих складових частин дистанційного навчання є його реалізація за допомогою використання інформаційних технологій, а саме системи управління навчанням, створеної для розробки, управління та поширення навчальних матеріалів онлайн із забезпеченням спільного доступу багатьох користувачів як правило через Internet.

На даний момент створена велика кількість різноманітних систем дистанційного навчання як з відкритим кодом, що є як правило безкоштовні повністю чи частково, так і платних, широковживаних та вузькоспеціалізованих. Ознайомимось зосновними системами дистанційного навчання, які використовуються учбовими закладами по всьому світі.

ATutor — веб-орієнтована система керування навчанням (Learning Management System, LMS)[6]. Програмний продукт є простим у встановленні, налаштуванні та підтримці для системних адміністраторів; викладачі (інструктори) можуть досить легко створювати та переносити навчальні матеріали та запускати свої онлайн-курси. А оскільки система є модульна, тобто складається з окремих функціональних одиниць — модулів, то вона відкрита для модернізації і розширення функціональних можливостей.

Програма розробляється та підтримується з 2001 року Ґреґом Ґеєм (Greg Gay), Джоелом Кроненберґом (Joel Kronenberg), Гайді Гейзелтон (Heidi Hazelton) із Дослідницького центру адаптивних технологій Університету Торонто (Adaptive Technology Resource Centre, University of Toronto). Система ATutor поширюється на основі GNU General Public License (GPL), яка, зокрема, дозволяє вільно використовувати, змінювати та доповнювати програму [6].

На рис. 2.1 зображено головну сторінку яку бачить користувач при вході в систему. Як бачимо тут є меню, перелік категорій контенту, так навігація по основним блокам таким як форум, вікі, презентації, календар тощо.

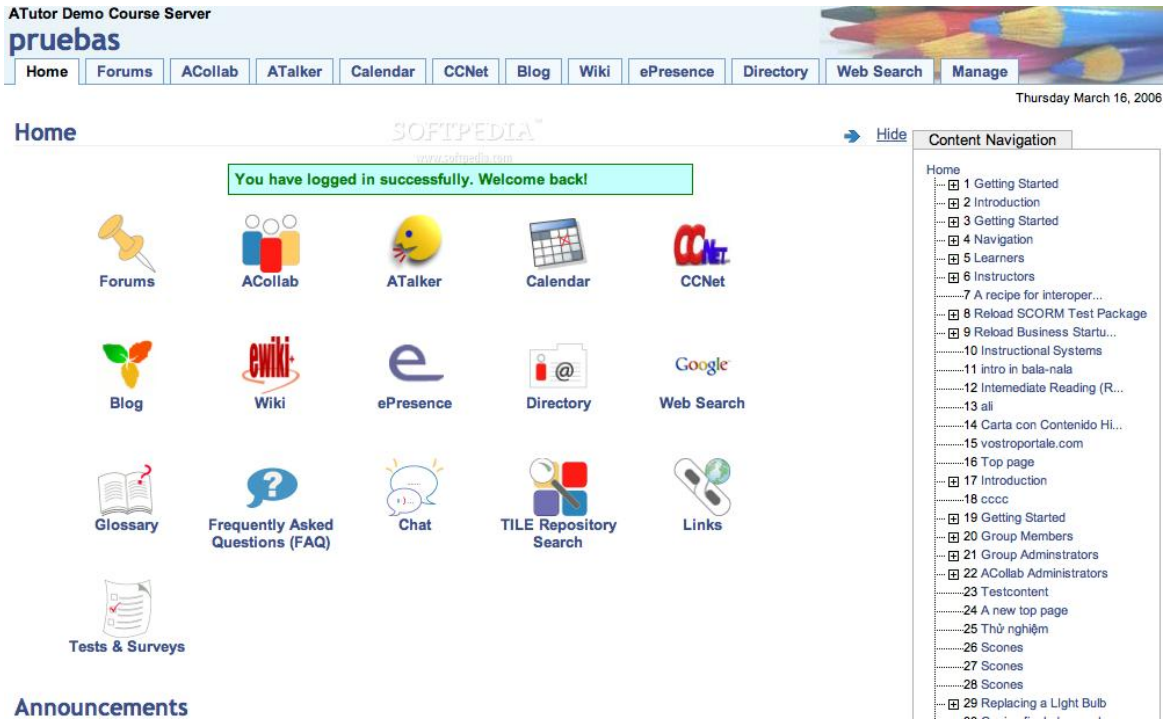


Рис 2.1 – Сторінка користувача ATutor

Learning Space (Lotus / IBM) - програмне навчальне середовище, яке об'єднує в собі можливості "класичного" навчання з сучасними інформаційними технологіями, заснованими на автоматизації взаємодії викладача зі студентами.

Learning Space дає можливість вчитися і викладати в режимі використанняматеріалів курсів у різнийчас і брати участь в on-line заняттях в режимі реального часу. Користувач може створювати зміст курсу в будь-яких додатках і потім розміщувати створений матеріал в Learning Space. Програма має гнучку систему редагування та адміністрування курсу, дозволяє вибирати різні режими викладання і стежити за поточними результатами роботи учнів. Learning Space робить навчання незалежним від місця знаходження його учасників. Для участі в навчальному процесі необхідно мати тільки доступ в Інтернет [7].

На рис. 2.2 Відображено головне вікно для користувача. Воно складається з таких елементів як блок навігації, що має кнопки переходу до каталогу курсів, курсів користувача, також з лівої сторони розміщено меню в якому можна перейти до розкладу, перегляду прогресу, інформації та інші сторінки.

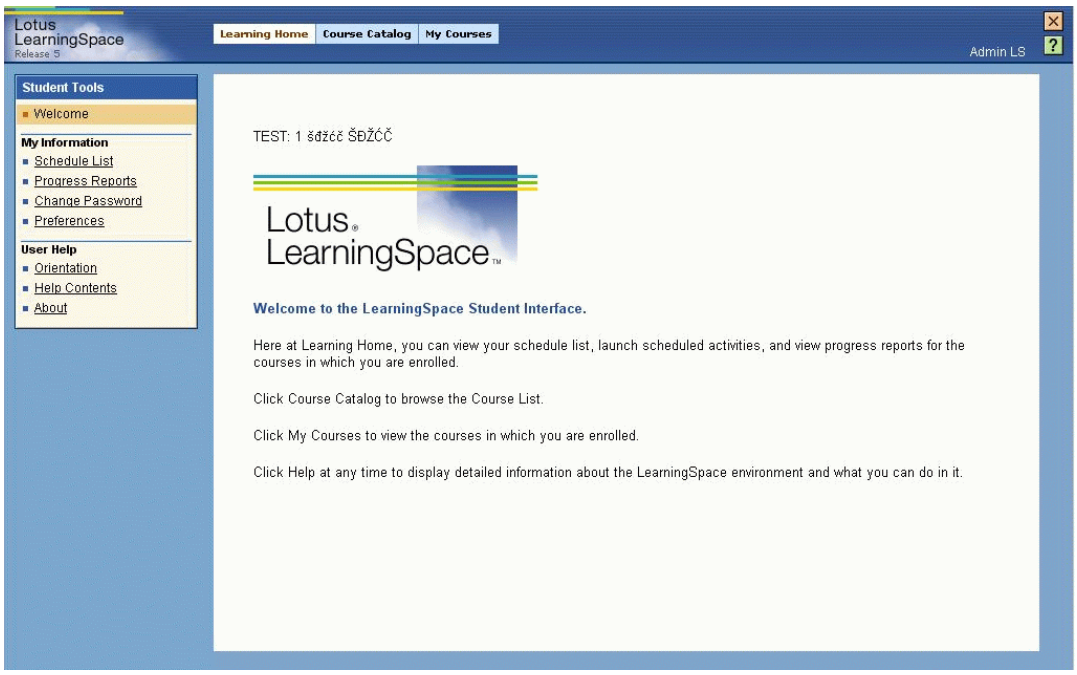


Рис. 2.2 – Головне вікно користувача

Основні можливості системи:

- Розподіленість - можливість вчитися в будь-якому місці і в будь-який час;
- Гнучкість - можливість навчання в потрібному вам темпі;
- Групове співробітництво - можливість індивідуального або групового навчання;
- Вибір викладачів - можливість навчання у досвідчених експертів;
- Простота - призначений для користувача інтерфейс допомагає легко переходити від одного модуля до іншого;
- Практичний досвід - курси засновані не на "лекціях", а на практичних заняттях;
- Доступ до додаткових матеріалів - навчання проходить з використанням багатих і гнучких можливостей електронного середовища;
- Безпека - безпечні віртуальні області для ведення дискусій, отримання оцінок і розміщення приватних оголошень;
- Групові завдання - з використанням методики групового авторства;
- Багатозадачність - участь в організованих дискусіях з багатьох потоків.

Розглянувши ці системи дистанційного навчання ми бачимо, що кожна з них має як свої переваги так і деякі недоліки, тому неможливо чітко вибрати найкращу систему. Кожна система проектувалась з урахуванням покращення навчального процесу і зручності в користуванні як для педагогів так і студентів. Всі системи мають засоби, що дозволяють вести статистику успішності чи виконання робіт, можливість переписування чи вести дискусії в форматі форуму.

Отже, як бачимо немає чіткої відповіді на питання – яка система найкраща, тому при обранні системи потрібно в першу чергу орієнтуватись на потреби і можливості навчального закладу.

2.3. Програми – тренажери в дистанційній освіті

Великі можливості щодо формування творчої особистості фахівця містять інтерактивні методи навчання. Такі методи реалізують програми тренажери, які дозволяють користувачеві закріпити теоретичні матеріали на практиці, отримати нові знання в інтерактивній формі при взаємодії з програмою.

Тренажер реалізуєнаступні аспекти:

- оцінка обсягу наданого матеріалу;
- врахування як рівня підготовки так і мотиву навчання;
- встановлення алгоритму виконання певної роботи.

Прийнято, що тренажер складається з декількох дій чи певних кроків. На кожному етапі студент перевіряється на певні знання, при правильних відповідях він отримує можливість рухатись далі і також завершення роботи зпрограмою. На кожному етапі виконання роботи студент має можливість задати питання викладачеві або звертатись до матеріалів лекції, які також можуть бути інтегровані в тренажер. Об'єм завдань тренажерамає бути достатній для забезпечення роботою групу студентів.

3 ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритмізація задачі за темою роботи.

Програмна реалізація програмного забезпечення з теми «Алгоритми сортування», що навчає алгоритмам сортування передбачає наявність умови задачі на кожному кроці виконання тренінгу, а також виконання тестових завдань (теоретичного) і практичного плану.

Розлянемо кожен крок.

Крок 1.

На экрані форма реєстрації, де треба заповнити поля, для того щоб почати роботу с тренажером, потрібно заповнити ці поля, та натистути кнопку «Зареєструватися»

Крок 2.

Після успішної реєстрації відкриється сторінка з усіма тестами.

Крок 3.

Після натискання «Сортування вибором» або «Сортування бульбашкою», або «Сортування вставками», або «Швидке сортування» відкривается сторінка з питаннями, які потрібно пройти:

Що таке змінна?

a) пойменована, або адресується іншим способом область пам'яті, адреса якої можна використовувати для здійснення доступу до даних і змінювати значення в ході виконання програми.
b) пойменована, або адресується іншим способом область пам'яті, адреса якої можна використовувати для здійснення доступу до даних і НЕ змінювати значення в ході виконання програми.
c) пойменована, або адресується іншим способом область пам'яті, яку можна використати лише в циклі.

Що таке масив?

a) змінна.
b) Структура даних.
c) цикл по змінних

Що таке цикл?

a) керуюча конструкція призначена для організації багаторазового виконання набору інструкцій.
b) Константа.
c) об'єкт.

Крок 4 Після натискання «Сортування злиттям» відкривается сторінка з питаннями, які потрібно пройти:

Що таке змінна?

a) пойменована, або адресується іншим способом область пам'яті, адресаякої можна використовувати для здійснення доступу до даних і змінювати значення в ході виконання програми.
b) пойменована, або адресується іншим способом область пам'яті, адреса якої можна використовувати для здійснення доступу до даних і НЕ змінювати значення в ході виконання програми.
c) пойменована, або адресується іншим способом область пам'яті, яку можна використати лише в циклі.

Що таке масив?

a) змінна.
b) Структура даних.
c) цикл по змінних

Що робить функція concat()?

a) керуюча конструкція призначена для організації багаторазового виконання набору інструкцій.
b) Константа.
c) об'єкт.

Крок 5.

Якщо відповіді на серію питань правильні, то відбувається перехід на сторінку з основним тестом, якщо відповідь не правильна, то відбувається перехід на сторінку з інформаційними відомостями де також буде посилання на повторне проходження тесту.

Крок 6.

На екрані код, де потрібно заповнити три поля, та дві кнопки «Правильна відповідь», яка стане активної тільки після хочаб однієї самостійної спроби заповнити поля, при натисканні на яку є можливість дізнатися правильну відповідь. «Відправити відповідь» при натисканні на яку, відбудеться перевірка відповіді та виведення на екран «Правильно» у разі правильної відповіді та «Не правильно» у разі не правильної відповіді.

Крок 7.

Після успішного виконання тесту кнопка на сторінкі з усіма тестами змінить свій колір з червого на зелений, що буде означити проходження тесту.

3.2. Розробка блок-схеми, яка підлягають програмуванню.

У цьому пункті бакалаврської роботи представлена блок-схема роботи програми, розроблена під час реалізації тренажера. На рис. 3.1 представлена блок-схема алгоритму роботи тренажера.

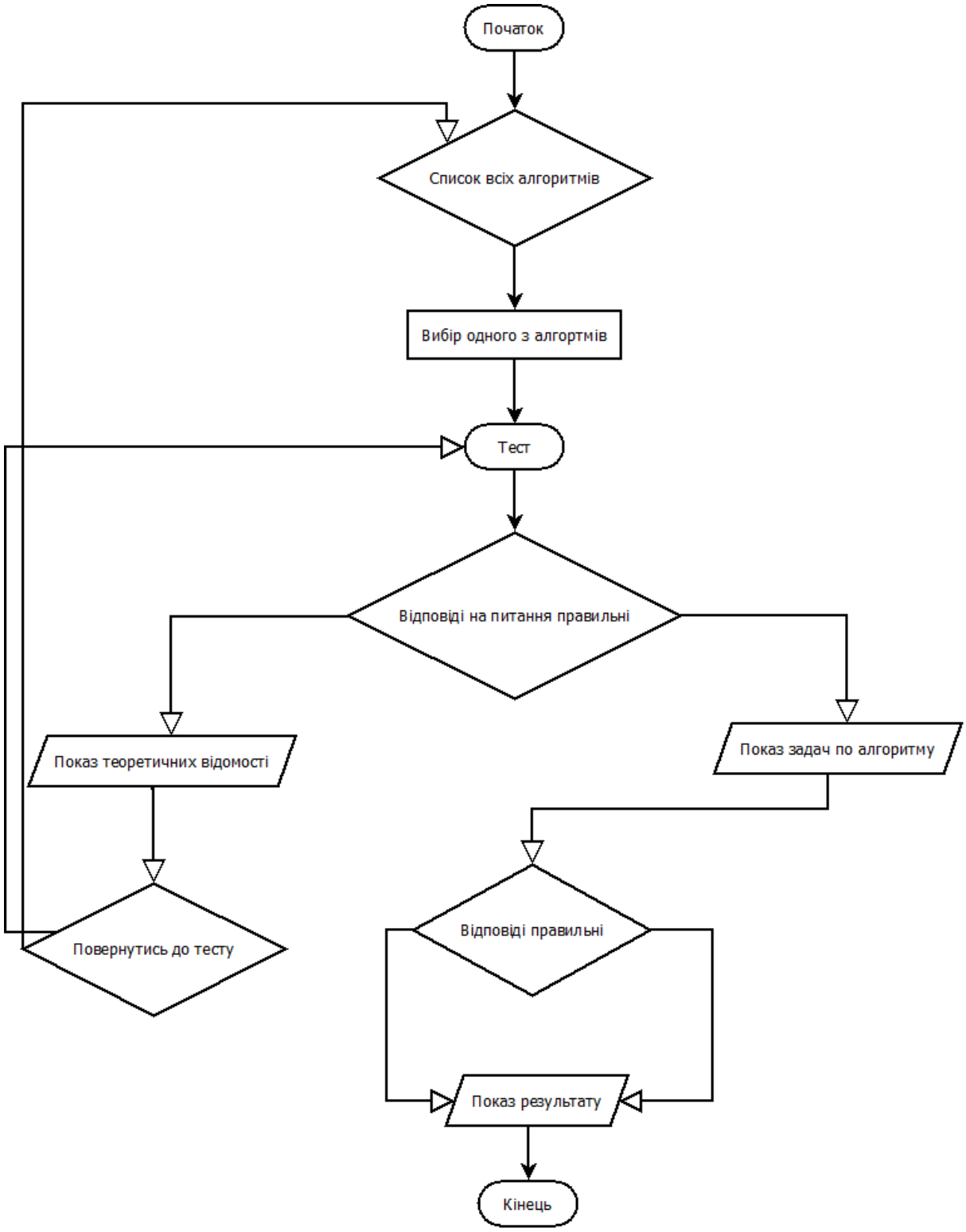


Рис. 3.1 – Блок-схема алгоритму роботи тренажера

3.3. Алгоритми сортування.

Сортування обміном або сортування бульбашкою є простим алгоритмом сортування. Алгоритм працює таким чином — у поданому наборі даних (списку чи масиві) порівнюються два сусідні елементи. Якщо один з елементів, не відповідає критерію сортування (є більшим, або ж, навпаки, меншим за свого сусіда), то ці два елементи міняються місцями. Прохід по списку продовжується доти, доки дані не будуть відсортованими. Алгоритм отримав свою назву від того, що процес сортування за ним нагадує поведінку бульбашок повітря у резервуарі з водою. Оскільки для роботи з елементами масиву він використовує лише порівняння, це сортування на основі порівнянь.

Сортування вибором — простий алгоритм сортування лінійного масиву, на основі вставок. Має ефективність n^2, що робить його неефективним при сортування великих масивів, і в цілому, менш ефективним за подібний алгоритм сортування включенням. Сортування вибором вирізняється більшою простотою, ніж сортування включенням, і в деяких випадках, вищою продуктивністю.

Алгоритм працює таким чином:

— Знаходить у списку найменше значення
— Міняє його місцями із першим значеннями у списку
— Повторює два попередніх кроки, доки список не завершиться (починаючи з наступної позиції)

Сортування включенням — простий алгоритм сортування на основі порівнянь. На великих масивах є значно менш ефективним за такі алгоритми, як швидке сортування, пірамідальне сортування та сортування злиттям. Однак, має цілу низку переваг:

— простота у реалізації
— ефективний (зазвичай) на маленьких масивах

— ефективний при сортуванні масивів, дані в яких вже непогано відсортовані: продуктивність рівна O($n + d$), де d — кількість інверсій
— на практиці ефективніший за більшість інших квадратичних алгоритмів (O(n^2)), як то сортування вибором та сортування бульбашкою: його швидкодія рівна $n^2/4$, і в найкращому випадку є лінійною

На кожному кроці алгоритму ми вибираємо один з елементів вхідних даних і вставляємо його на потрібну позицію у вже відсортованому списку доти, доки набір вхідних даних не буде вичерпано. Метод вибору чергового елементу з початкового масиву довільний; може використовуватися практично будь-який алгоритм вибору. Зазвичай (і з метою отримання стійкого алгоритму сортування), елементи вставляються за порядком їх появи у вхідному масиві.

Сортування злиттям — алгоритм сортування, в основіякого лежить принцип «Розділяй та володарюй».

В основі цього способу сортування лежить злиття двох упорядкованих ділянок масиву в одну впорядковану ділянку іншого масиву. Злиття двох упорядкованих послідовностей можна порівняти з перебудовою двох колон солдатів, вишикуваних за зростом, в одну, де вони також розташовуються за зростом. Якщо цим процесом керує офіцер, то він порівнює зріст солдатів, перших у своїх колонах і вказує, якому з них треба ставати останнім у нову колону, а кому залишатися першим у своїй. Так він вчиняє, поки одна з колон не вичерпається — тоді решта іншої колони додається до нової.

Під час сортування в дві допоміжні черги з основної поміщаються перші дві відсортовані підпослідовності, які потім зливаються в одну і результат записується в тимчасову чергу. Потім з основної черги беруться наступні дві відсортовані підпослідовності і так доти, доки основна черга не стане порожньою. Після цього послідовність з тимчасової черги переміщається в основну чергу. І знову продовжується сортування злиттям двох відсортованих підпослідовностей. Сортування триватиме доти, доки довжина відсортованої підпослідовності не стане рівною довжині самої послідовності.

Швидке сортування — алгоритм сортування, розроблений Тоні Гоаром, який не потребує додаткової пам'яті і виконує у середньому операцій. Однак, у найгіршому випадку робить порівнянь. Позаяк алгоритм використовує дуже прості цикли і операції, він працює швидше за інші алгоритми, що мають таку ж асимптотичну оцінку складності. Наприклад, зазвичай більш ніж удвічі швидший порівняно з сортуванням злиттям.

Ідея алгоритму полягає в переставлянні елементів масиву таким чином, щоб його можна було розділити на дві частини і кожний елемент з першої частини був не більший за будь-який елемент з другої. Впорядкування кожної з частин відбувається рекурсивно. Алгоритм швидкого сортування може бути реалізований як у масиві, так і в двозв'язному списку.

Швидке сортування є алгоритмом на основі порівнянь і не є стабільним.

4 ПРАКТИЧНА ЧАСТИНА

4.1. Опис процесу програмної реалізації

Оскільки, тренажер було вирішено створювати за допомогою HTML, CSS, JavaScriptбуло обрано середовище VisualStudioCodeЦе найпопулярніше середовище розробки і більше орієнтоване на програмування, ніж на дизайн. Останній буде створений за допомогою JavaScriptбез застосування бібліотек, На рис 4.1. зображено створення нового проекту середовищі VisualStudioCode.

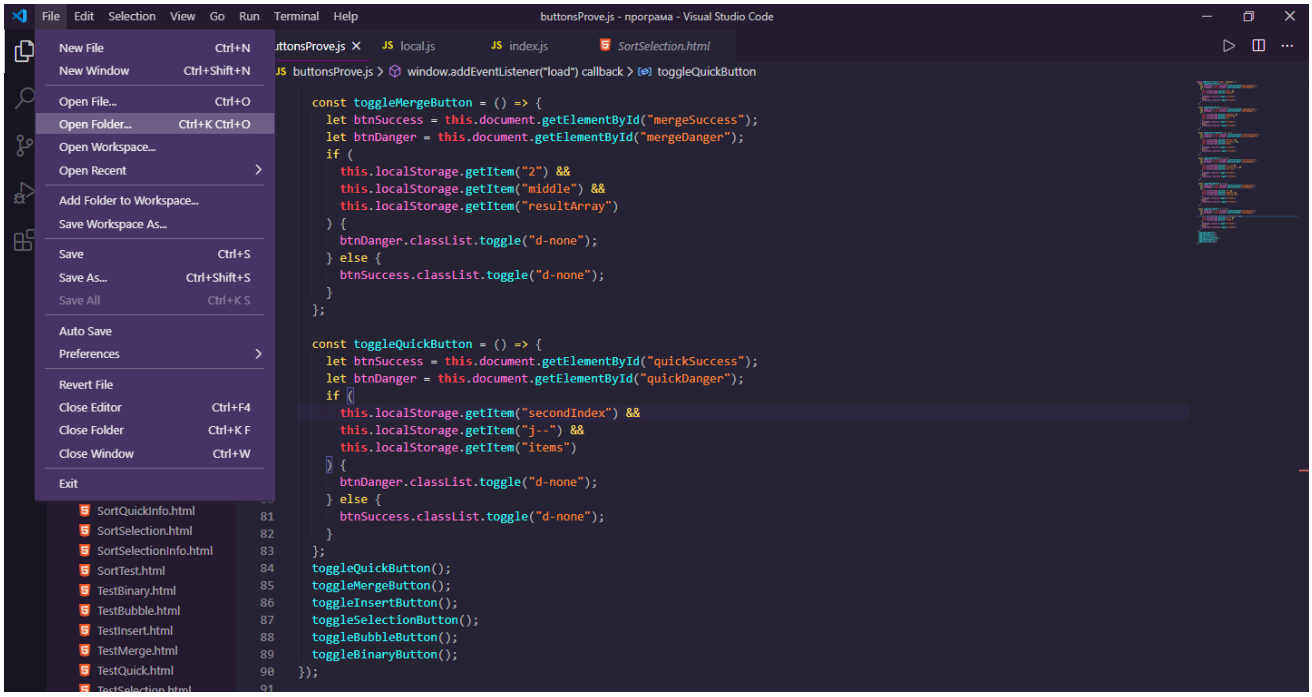


Рисунок 4.1. – Вікно середовища створення VisualStudioCode

Під час написання програми були реалізовані деякі функції, кожна з яких відповідала за певний крок алгоритму.

Для реєстрації користувача у localStorageбула створена така функція:

```
window.addEventListener("load", function () {
  if (
    this.localStorage.getItem("name") &&
    this.localStorage.getItem("password")
  ) {
    this.document.location.href = "List.html";
  }

  const submitRegister = (name, password) => {
    if (
```

```
    this.localStorage.getItem("name") === null &&
    this.localStorage.getItem("password") === null
   ) {
    this.localStorage.setItem("name", name);
    this.localStorage.setItem("password", password);
   }
  };

  if (this.document.getElementById("submitReg")) {
   let btn = this.document.getElementById("submitReg");
   btn.addEventListener("click", () => {
    let name = this.document.getElementById("nameReg").value;
    let password = this.document.getElementById("passwordReg").value;
    submitRegister(name, password);
    this.document.location.href = "List.html";
   });
  }
});
```

Для перевірки правильності відповіді користувача на тестові питання було створено таку функцію:

```
function arrSr(arr1, arr2) {
  if (arr1.length !== arr2.length) {
    return false;
  }
  for (let i = 0; i < arr1.length; i++) {
    if (arr1[i] !== arr2[i].getAttribute("data-val")) {
      return false;
    }
  }
  return true;
}
```

Для перевірки правильності відповіді користувача на завдання з заповненням порожніх комірок було написану таку функцію:

```
function inputTasksSubmit(input, btn, rid, lid, arr) {
    let inputs = this.document.querySelectorAll(input);
    let codeBtn = this.document.getElementById(btn);
    let rightId = this.document.getElementById(rid).getAttribute("id");
    let leftId = this.document.getElementById(lid).getAttribute("id");
```

```
    let array = arr;

    if (codeBtn) {
      codeBtn.addEventListener("click", () => {
        let inputsArr = [];
        for (let i = 0; i < inputs.length; i++) {
          if (inputs[i].value !== "" && inputs[i].value !== null) {
            inputsArr.push(inputs[i].value);
          }
        }
        if (inputsArr.length !== array.length) {
          $("#" + leftId).modal();
          return;
        } else {
          if (
            inputsArr[0] == array[0] &&
            inputsArr[1] == array[1] &&
            inputsArr[2] == array[2]
          ) {
            this.localStorage.setItem(array[0], array[0]);
            this.localStorage.setItem(array[1], array[1]);
            this.localStorage.setItem(array[2], array[2]);
            $("#" + rightId).modal();
          } else {
            $("#" + leftId).modal();
}
        }
      });
    }
  }
```

За домогою цих функцій було створено весь алгоритм роботи тренажеру. Повний код програми наведено у додатку А.

4.2. Технології реалізації тренажеру

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. [1]

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу. [1]

Мова JavaScript використовується для:

— написання сценаріїв веб-сторінок для надання їм інтерактивності;
— створення односторінкових веб-застосунків (React, AngularJS, Vue.js);
— програмування на стороні сервера (Node.js);
— стаціонарних застосунків (Electron, NW.js);
— мобільних застосунків (React Native, Cordova);
— сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
— всередині PDF-документів тощо.

ФРЕЙМВОРКИ ТА БІБЛІОТЕКИ

Bootstrap—вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів. jQuery— набір функцій JavaScript, що фокусується на взаємодії JavaScript і HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX. Розробка jQuery ведеться командою добровольців на пожертвування. [4]

СЕРЕДОВИЩЕ РОЗРОБКИ

Visual Studio Code — засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS.[8]

4.3. Опис програми.

На рисунку 4.2 представлений скріншот зовнішнього вигляду програми.

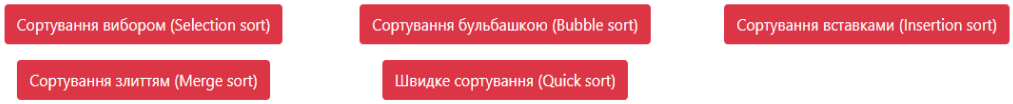


Рисунок 4.2 – Панель доступних тестів сторінка

Функціонал цього вікна було створено за допомогою цього коду:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<link rel="stylesheet" href="css/bootstrap.min.css">
<link rel="stylesheet" href="css/index.css">
<title>Список алгоритмів</title>
</head>
<body>

<div class="container">
<div class="row">
<div class="col-lg-4 text-center algo"><a class="btn btn-success" href="TestBinary.html">Бінарний пошук</a></div>
```

```
<div class="col-lg-4 text-center algo"><a class="btn btn-success" href="SortBubbleTest.html">Сортування бульбашкою</a></div>
</div>
</div>
<script src="js/jquery.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/index.js"></script>
</body>
</html>
```

На даному скріншоті зображене вікно програми, що містить два посилання надва алгоритми «Сортування бульбашкою», «Сортування вибором», «Сортування вставками», «Сортування злиттям», «Швидке сортування».

При натисканні на один з алгоримтів відкривається перехід на панель тестування для цього алгоритму.

Після натискання кнопки на одне з посилань запускається проходження тесту для вибраного алгоритму як на рисунку 4.3.

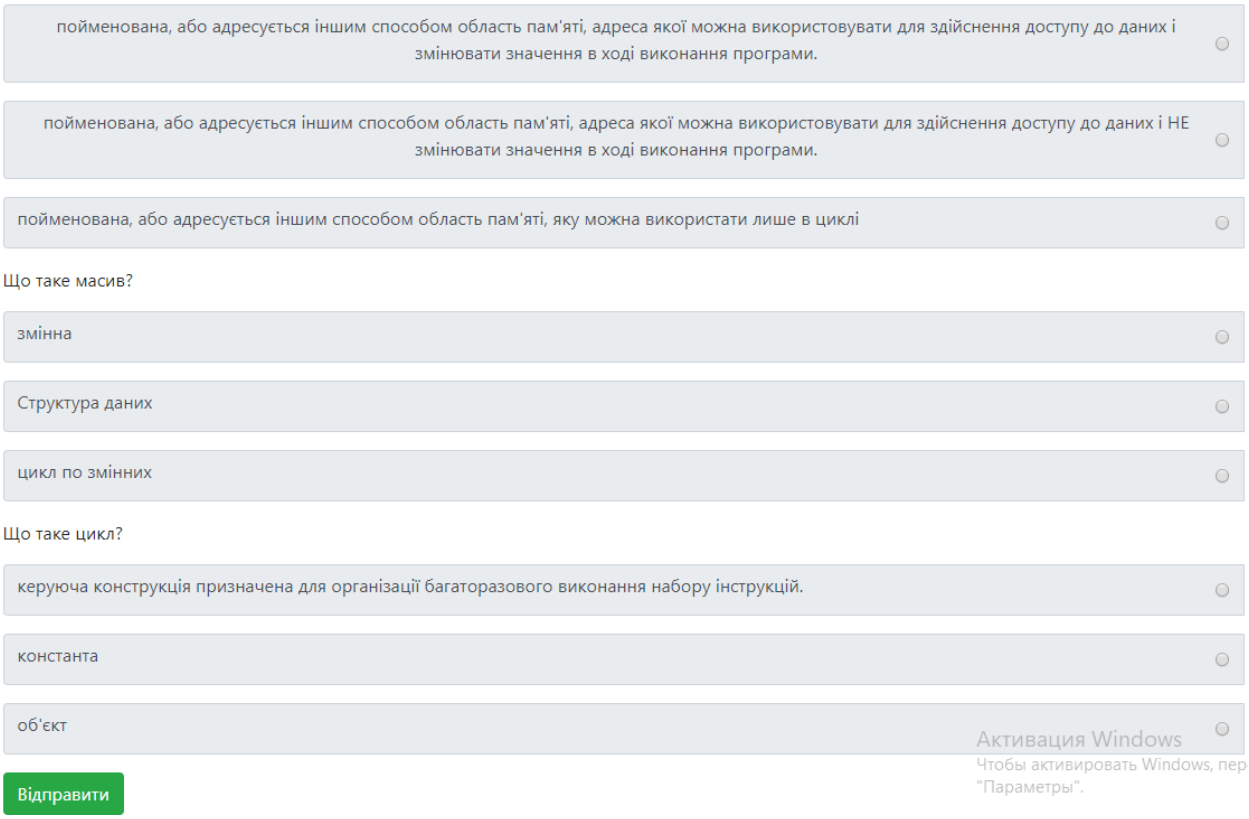


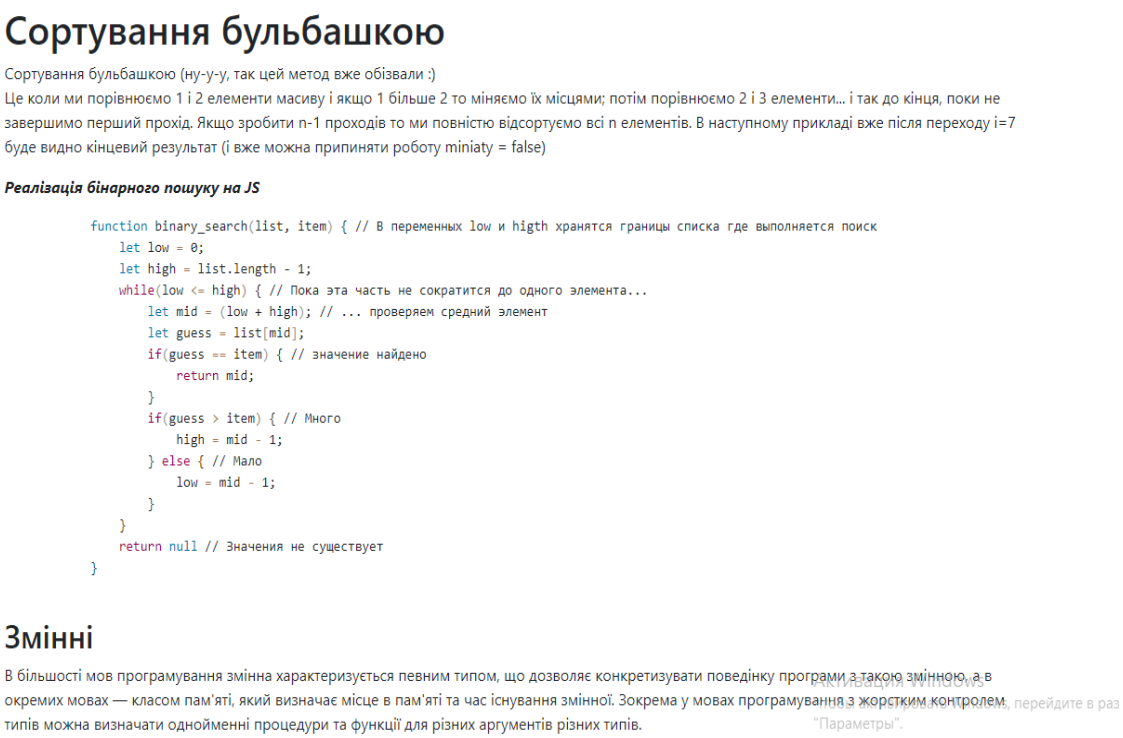
Рисунок 4.3 – Вибір варіанту відповіді

Процес проходження тесту реалізовується наступним функціоналом:

```
function rigthRoute( btn, classN, strRoute1, strRoute2 ) {

    let mainBtn = this.document.getElementById( btn );
    arr = [];

    if ( mainBtn ) {

      mainBtn.addEventListener('click', () => {

        let answerArr = this.document.querySelectorAll( classN );

        if ( answerArr.length > 0 ) {
```

```
            answerArr.forEach( ( item ) => {
                if ( item.checked ) {
                    arr.push( item );
                }
            });
        }

        if ( arrSr( answerRightArr, arr ) ) {
            if ( answerArr.length > 0 ) {
                this.document.location.href = strRoute1;
            }
        } else {
            this.document.location.href = strRoute2
        }
        arr = [];

    });
  }
}
```

Після вибору відповіді, при натисненні на кнопку «Відповісти» у випадку правильної відповіді відбувається перехід до задач по цьому алгоритму, якщо відповідь не правильна відбудеться перехід до теоретичної довідки як на рисунку 4.4.



Сортування бульбашкою

Сортування бульбашкою (ну-у-у, так цей метод вже обізвали :)
Це коли ми порівнюємо 1 і 2 елементи масиву і якщо 1 більше 2 то міняємо їх місцями; потім порівнюємо 2 і 3 елементи... і так до кінця, поки не завершимо перший прохід. Якщо зробити n-1 проходів то ми повністю відсортуємо всі n елементів. В наступному прикладі вже після переходу i=7 буде видно кінцевий результат (і вже можна припиняти роботу miniaty = false)

Реалізація бінарного пошуку на JS

```
function binary_search(list, item) { // В переменных low и higth хранятся границы списка где выполняется поиск
    let low = 0;
    let high = list.length - 1;
    while(low <= high) { // Пока эта часть не сократится до одного элемента...
        let mid = (low + high); // ... проверяем средний элемент
        let guess = list[mid];
        if(guess == item) { // значение найдено
            return mid;
        }
        if(guess > item) { // Много
            high = mid - 1;
        } else { // Мало
            low = mid - 1;
        }
    }
    return null // Значения не существует
}
```

Змінні

В більшості мов програмування змінна характеризується певним типом, що дозволяє конкретизувати поведінку програми з такою змінною, а в окремих мовах — класом пам'яті, який визначає місце в пам'яті та час існування змінної. Зокрема у мовах програмування з жорстким контролем типів можна визначати однойменні процедури та функції для різних аргументів різних типів.

Рисунок 4.4 – Теоретична довідка

На рисунку 4.3. зображено теоретичне питання з вибором варіанту відповіді, є ще один тип практичного завдання, а саме:практичне завдання з ручним внесенням відповіді на рисунку4.4.

Рисунок 4.5 – Практичне завдання з ручним внесенням

Після відповіді на завдання, відобразиться результат «Правильно» у випадку правильної відповіді и «Не правильно» у випадку не правильної відповіді. При правильному проходженні завдання на рисунку 4.6. та при не провальному на рисунку 4.7.

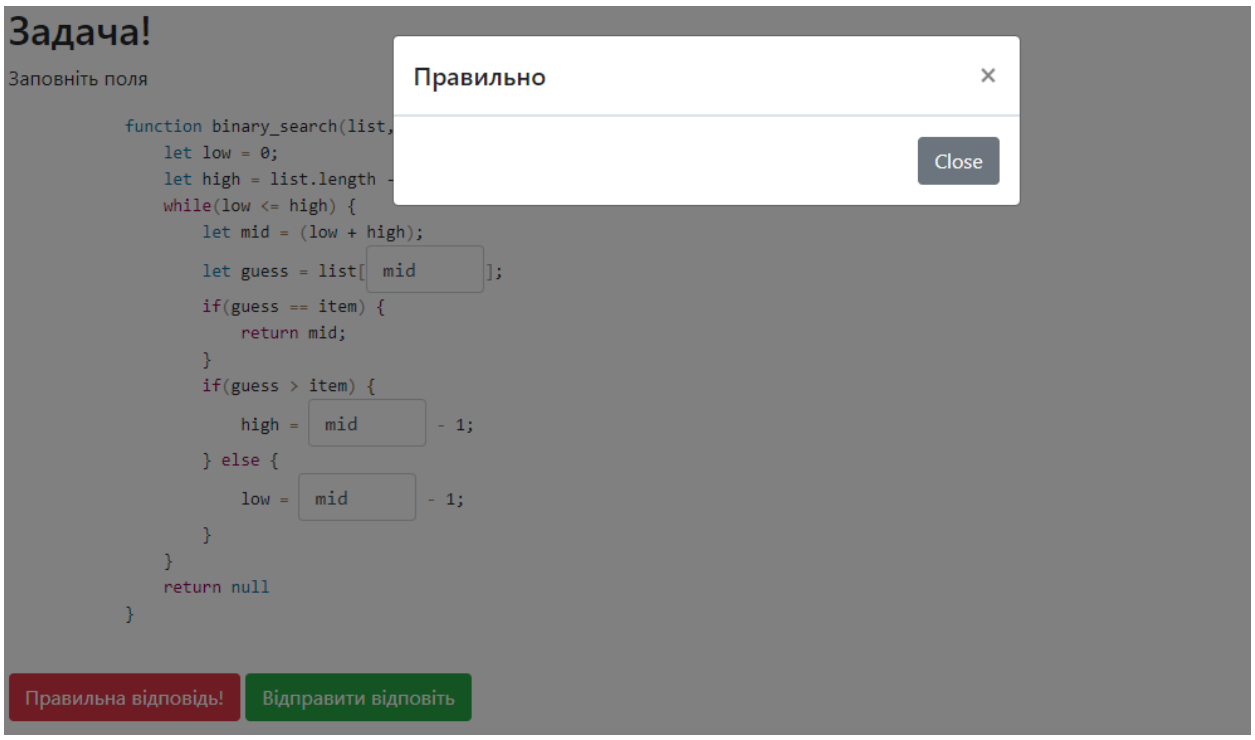


Рисунок 4.6 – Правильнерішеннязадачі

Код виводу результату:

```
function inputTasksSubmit ( input, btn, rid, lid, arr ) {
    let inputs = this.document.querySelectorAll( input );
    let codeBtn = this.document.getElementById( btn );
    let rightId = this.document.getElementById( rid ).getAttribute('id');
    let leftId = this.document.getElementById( lid ).getAttribute('id');
    let array = arr;

    if (codeBtn) {
      codeBtn.addEventListener('click', () => {

        let inputsArr = [];
        for (let i = 0; i < inputs.length; i++) {
```

```
                if (inputs[i].value !== "" && inputs[i].value !== null) {
                    inputsArr.push(inputs[i].value);
                }
            }
            this.console.log(inputsArr);
            if (inputsArr.length !== array.length) {
                $('#' + leftId).modal();
                return;
            } else {

                if (inputsArr[0] == array[0] && inputsArr[1] == array[1]
&& inputsArr[2] == array[2]) {
                    $('#' + rightId).modal();
                } else {
                    $('#' + leftId).modal();
                }

            }

        });
    }
}
function rightModal ( btn, input, rigth, left, rightAns ) {
    let btnSubmitAns = this.document.getElementById( btn );
    let inputAns = this.document.getElementById( input );
```

```
    let rigthId = this.document.getElementById( rigth ).getAttribute('id');
    let leftId = this.document.getElementById( left ).getAttribute('id');
    let rightAnswer = rightAns;


    if ( btnSubmitAns ) {
        btnSubmitAns.addEventListener('click', () => {
            if (inputAns.value == rightAnswer) {
                $('#' + rigthId).modal();
            } else {
                $('#' + leftId).modal();
            }
        });
    }
}
```

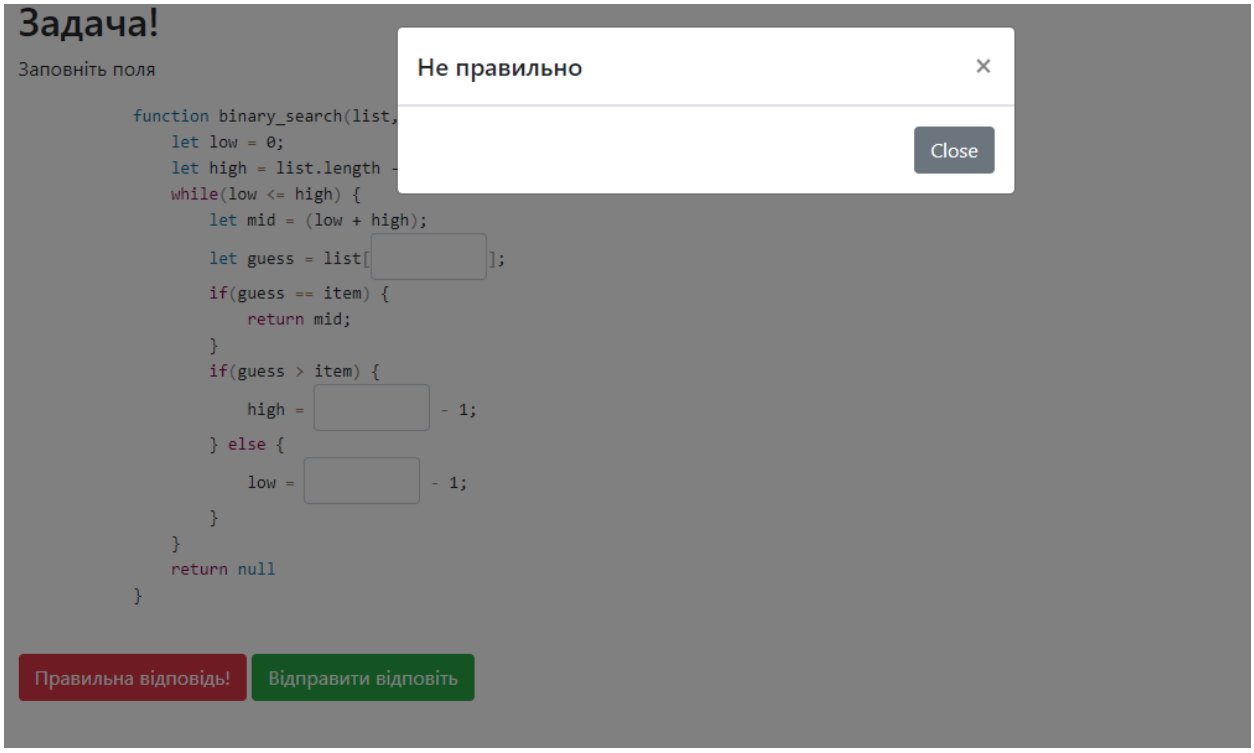


Рисунок 4.7 – Не правильнерішеннязадачі

4.4. Інструкція користувача

Щоб почати роботу потрібно знайти сайт за посиланням http://www.diplom123.ho.ua/

Для того щоб почати роботу з програмаю треба увійти в систему за посиланням http://www.diplom123.ho.ua/ як зображено на рисунку 4.8

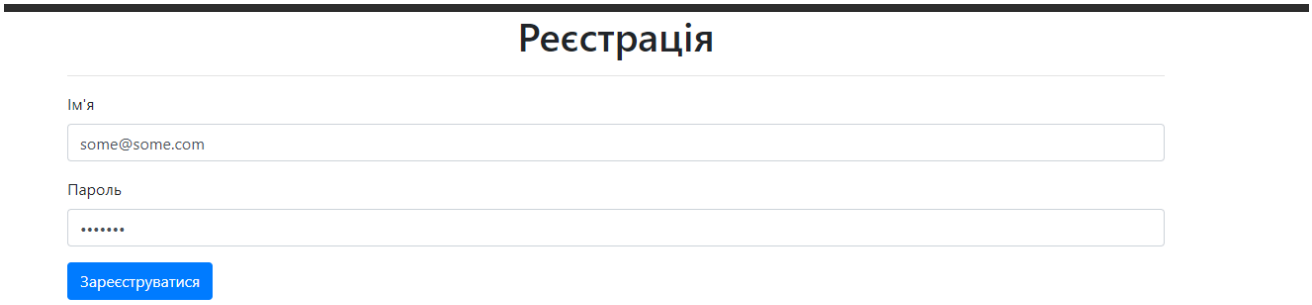


Рисунок 4.8. Форма реєстрації.

Після чого відкриется сторінка з усіма тестами як зображено на рисунку 4.9

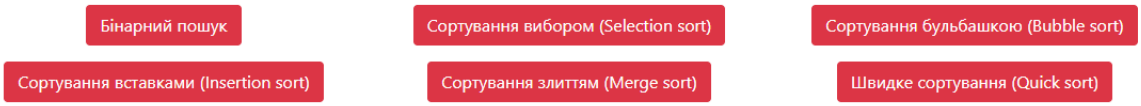


Рисунок 4.8. Список тестів.

Після вибору тесту, необхідно пройти пробне тестування, яке віявить чи може користувач починати основний тест, пробне тестування зображено на рисунку 4.9

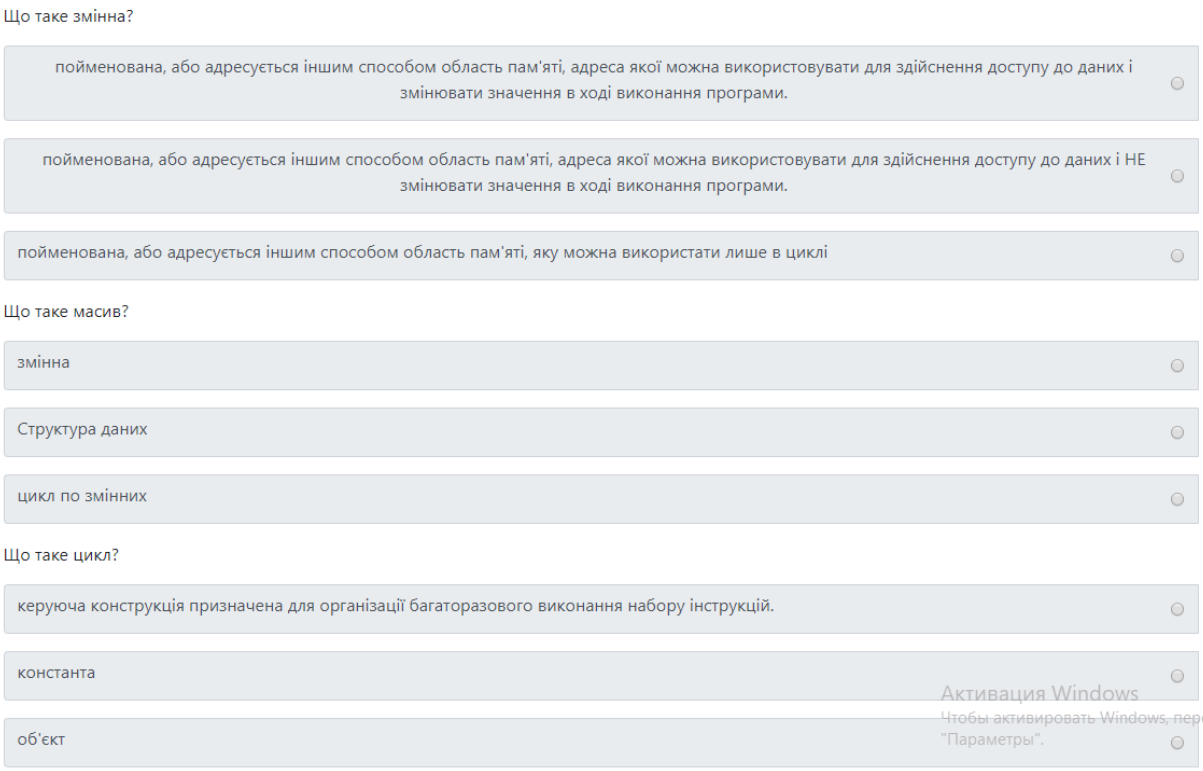


Рисунок 4.9. Пробне тестування

У разі не проходження пробного тестування відкриется сторінка з информаційними відомостями, як на рисунку 4.10.

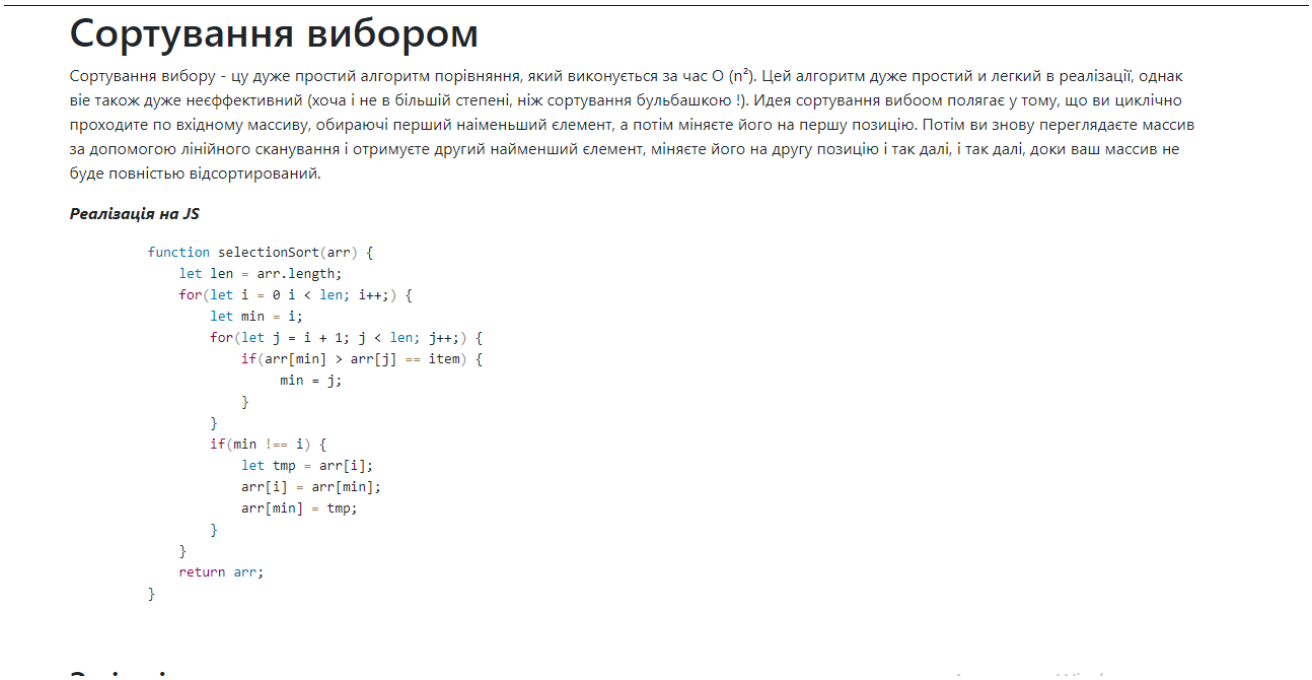


Рисунок 4.9. Інформаційні відомості.

У разі проходження пробного тестування відкриеться сторінка з основним тестом, як на рисунку 4.10.

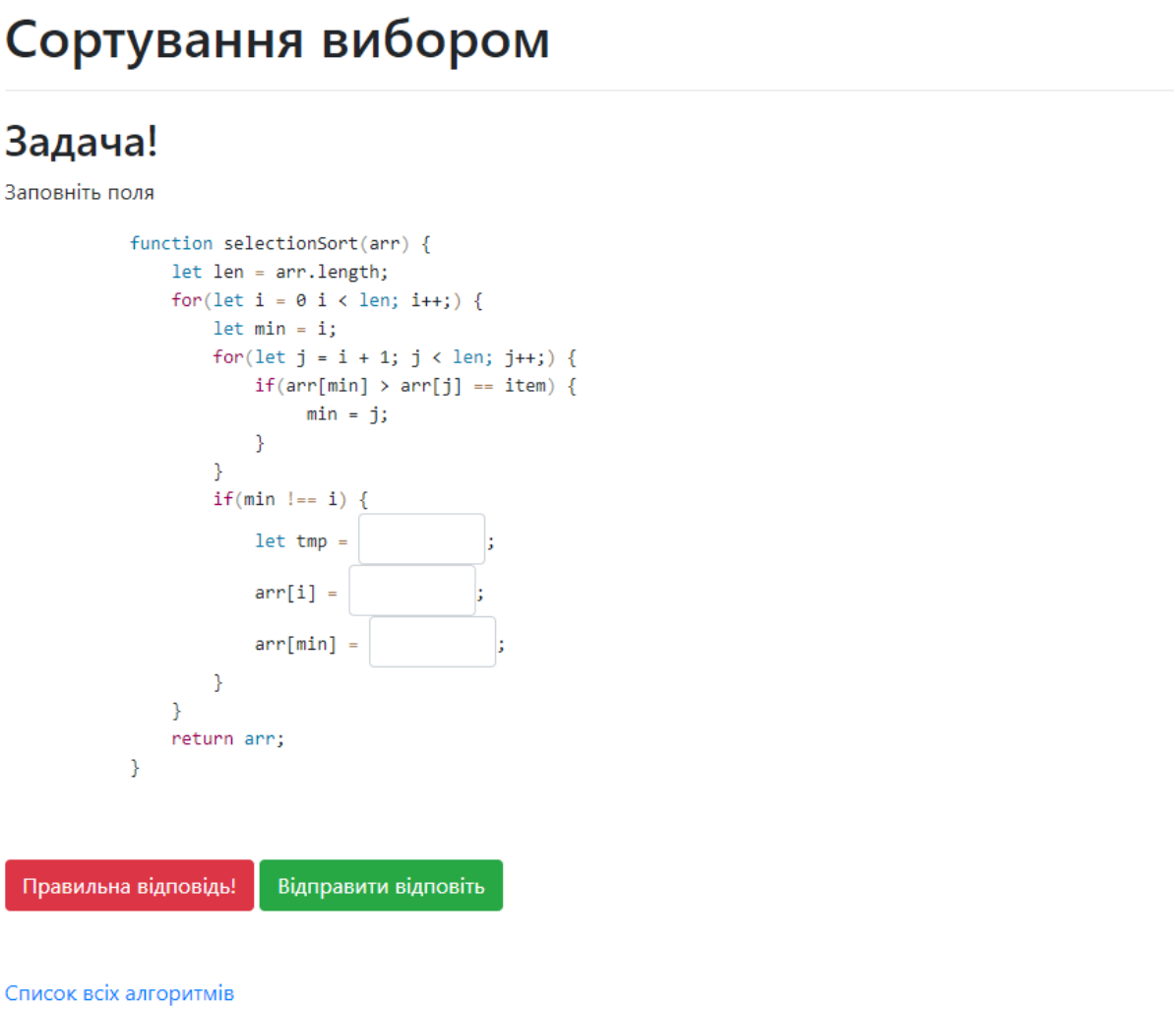


Рисунок 4.10. Основний тест.

В основному тесті необідно буде заповнити поля та натистути кнопку «Відправити відповідь» після чого відбудеться перевірка правильності відповіді та виведеться модельне вікно з текстом «Правильно» у разі правильної відповіді та «Не правильно» у разі не правильної відповіді.

Далі треба натистути на текст «Список всіх алгоритмів» и у разі проходження тесту кнопка з назвою тесту стане зеленою як на рисунку 4.11.

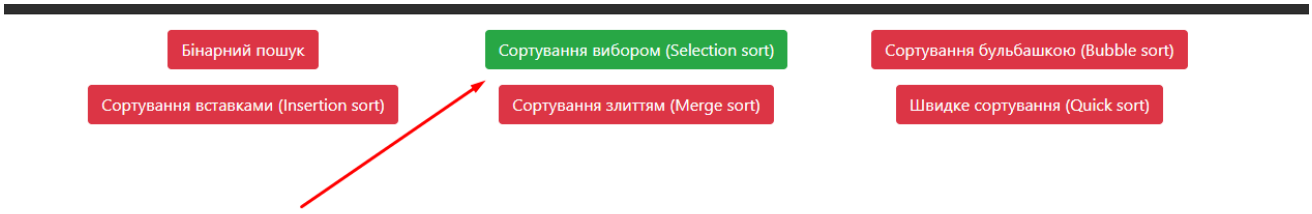


Рисунок 4.11. Пройдений тест

ВИСНОВКИ

Тренажери в дистанційній освіті займають одне з важливих місць, тому їх кількість потрібно збільшувати, щоб забезпечити якомога більше дисциплін ними, а якість покращувати, щоб користувач отримував максимум користі від їх використання.

Тренажери допомагають студентам самостійно покращувати й перевіряти свої навички і теоретичні знання з певних дисциплін. Також віртуальні тренажери можуть бути застосовані не тільки під час самостійних робіт, але й при проведені лабораторних занять.

При виконанні бакалаврської проекту було розроблено програмне забезпечення з теми «Алгоритми сортування», зокрема розглянуті алгоритми сортування бульбашкою, сортування вибором, сортування вставками», швидке сортування.

Для створення розробки було використано HTML5, CSS3, JavaScript.

В навчальному тренажері реалізовано наступні завдання:

- Автоматизована перевірка та виведення на екран помилки;
- Вводити відповідь з клавіатури.

Створений програмний продукт дозволить студентам перевіряти свої знання, а викладачам перевіряти студентів під час контрольних робіт.

У майбутньому планується подальша розробка даного проекту, а саме – детальний звіт по пройденим тестам, створення профілів студентів, щоб зберігати їх прогрес на протязі курсу. Також планується виведення проекту на хмарний сервіс для доступу до сайту, щоб всі мали змогу їм скористуватися.

.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. JavaScript — Вікіпедія [Електронний ресурс]. – 2020 – https://uk.wikipedia.org/wiki/JavaScript – Дата доступу: черв.2020. – Назва з екрану.
2. HTML5 — Вікіпедія [Електронний ресурс]. – 2020 – https://uk.wikipedia.org/wiki/HTML– Дата доступу: черв.2020. – Назва з екрану.
3. CSS3 — Вікіпедія [Електронний ресурс]. – 2020 – https://uk.wikipedia.org/wiki/CSS– Дата доступу: черв.2020. – Назва з екрану.
4. Bootstrap — Вікіпедія [Електронний ресурс]. – 2020. – https://uk.wikipedia.org/wiki/Bootstrap– Дата доступу: груд. 2020. – Назва з екрану.
5. Алгоритм сортування — Вікіпедія [Електронний ресурс]. – 2020. – https://uk.wikipedia.org/wiki/Алгоритм_сортування – Дата доступу: груд. 2020. – Назва з екрану.
6. ATutor — Вікіпедія [Електронний ресурс]. – 2020. – https://uk.wikipedia.org/wiki/ ATutor – Дата доступу: жовт. 2020. – Назва з екрану.
7. ATutor User Documentation. [Електронний ресурс]. – Режим доступу: http://help.atutor.ca/general/.
8. Visual Studio Code, Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code – Назва з екрану (дата звернення 10.4.2020).

ДОДАТОК А

```
window.addEventListener("load", function () {
 const toggleBinaryButton = () => {
  let btnSuccess = this.document.getElementById("binarySuccess");
  let btnDanger = this.document.getElementById("binaryDanger");
  if (
   this.localStorage.getItem("mid") &&
   this.localStorage.getItem("binary")
  ) {
   btnDanger.classList.toggle("d-none");
  } else {
   btnSuccess.classList.toggle("d-none");
  }
 };

 const toggleBubbleButton = () => {
  let btnSuccess = this.document.getElementById("bubbleSuccess");
  let btnDanger = this.document.getElementById("bubbleDanger");
  if (
   this.localStorage.getItem("wasSwap") &&
   this.localStorage.getItem("endj") &&
   this.localStorage.getItem("j")
  ) {
   btnDanger.classList.toggle("d-none");
  } else {
   btnSuccess.classList.toggle("d-none");
  }
 };

 const toggleSelectionButton = () => {
  let btnSuccess = this.document.getElementById("selectionSuccess");
  let btnDanger = this.document.getElementById("selectionDanger");
  if (
   this.localStorage.getItem("arr[i]") &&
   this.localStorage.getItem("arr[min]") &&
   this.localStorage.getItem("tmp")
  ) {
   btnDanger.classList.toggle("d-none");
  } else {
   btnSuccess.classList.toggle("d-none");
```

```
  }
};

const toggleInsertButton = () => {
  let btnSuccess = this.document.getElementById("insertSuccess");
  let btnDanger = this.document.getElementById("insertDanger");
  if (
    this.localStorage.getItem("i - 1") &&
    this.localStorage.getItem("inputArr[j]") &&
    this.localStorage.getItem("[j + 1]")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleMergeButton = () => {
  let btnSuccess = this.document.getElementById("mergeSuccess");
  let btnDanger = this.document.getElementById("mergeDanger");
  if (
    this.localStorage.getItem("2") &&
    this.localStorage.getItem("middle") &&
    this.localStorage.getItem("resultArray")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleQuickButton = () => {
  let btnSuccess = this.document.getElementById("quickSuccess");
  let btnDanger = this.document.getElementById("quickDanger");
  if (
    this.localStorage.getItem("secondIndex") &&
    this.localStorage.getItem("j--") &&
    this.localStorage.getItem("items")
  ) {
    btnDanger.classList.toggle("d-none");
```

```
    } else {
      btnSuccess.classList.toggle("d-none");
    }
  };
  toggleQuickButton();
  toggleMergeButton();
  toggleInsertButton();
  toggleSelectionButton();
  toggleBubbleButton();
  toggleBinaryButton();
});

window.addEventListener("load", function () {
  if (
    !this.localStorage.getItem("name") &&
    !this.localStorage.getItem("password")
  ) {
    this.document.location.href = "register.html";
  }

  function arrSr(arr1, arr2) {
    if (arr1.length !== arr2.length) {
      return false;
    }
    for (let i = 0; i < arr1.length; i++) {
      if (arr1[i] !== arr2[i].getAttribute("data-val")) {
        return false;
      }
    }
    return true;
  }

  let answerRightArr = ["a", "b", "c"];

  function rigthRoute(btn, classN, strRoute1, strRoute2) {
    let mainBtn = this.document.getElementById(btn);
    arr = [];

    if (mainBtn) {
      mainBtn.addEventListener("click", () => {
```

```
    let answerArr = this.document.querySelectorAll(classN);

    if (answerArr.length > 0) {
      answerArr.forEach((item) => {
        if (item.checked) {
          arr.push(item);
        }
      });
    }

    if (arrSr(answerRightArr, arr)) {
      if (answerArr.length > 0) {
        this.document.location.href = strRoute1;
      }
    } else {
      this.document.location.href = strRoute2;
    }
    arr = [];
  });
 }
}

function rightModal(btn, input, rigth, left, rightAns) {
 let btnSubmitAns = this.document.getElementById(btn);
 let inputAns = this.document.getElementById(input);
 let rigthId = this.document.getElementById(rigth).getAttribute("id");
 let leftId = this.document.getElementById(left).getAttribute("id");
 let rightAnswer = rightAns;

 if (btnSubmitAns) {
  btnSubmitAns.addEventListener("click", () => {
    if (inputAns.value == rightAnswer) {
      this.localStorage.setItem("binary", rightAnswer);
      $("#" + rigthId).modal();
    } else {
      $("#" + leftId).modal();
    }
  });
 }
}
```

```
function showRightAnswer(btn, id) {
  let myBtn = this.document.getElementById(btn);
  let superId = this.document.getElementById(id).getAttribute("id");

  if (myBtn) {
    myBtn.addEventListener("click", () => {
      let confirnMass = this.prompt(
        "Ви упевнені що не в змозі вирішити задачу ? так | ні"
      );
      if (confirnMass == "так" || confirnMass == "т") {
        $("#" + superId).modal();
      } else if (confirnMass == "ні" || confirnMass == "н") {
        this.alert("Чудово намагайся далі !");
      }
    });
  }
}

function inputTasksSubmit(input, btn, rid, lid, arr) {
  let inputs = this.document.querySelectorAll(input);
  let codeBtn = this.document.getElementById(btn);
  let rightId = this.document.getElementById(rid).getAttribute("id");
  let leftId = this.document.getElementById(lid).getAttribute("id");
  let array = arr;

  if (codeBtn) {
    codeBtn.addEventListener("click", () => {
      let inputsArr = [];
      for (let i = 0; i < inputs.length; i++) {
        if (inputs[i].value !== "" && inputs[i].value !== null) {
          inputsArr.push(inputs[i].value);
        }
      }
      if (inputsArr.length !== array.length) {
        $("#" + leftId).modal();
        return;
      } else {
        if (
          inputsArr[0] == array[0] &&
```

```
          inputsArr[1] == array[1] &&
          inputsArr[2] == array[2]
        ) {
          this.localStorage.setItem(array[0], array[0]);
          this.localStorage.setItem(array[1], array[1]);
          this.localStorage.setItem(array[2], array[2]);
          $("#" + rightId).modal();
        } else {
          $("#" + leftId).modal();
        }
      }
    });
  }
}

// редиректы
//   rigthRoute("submit-c", ".some2", "Sort.html", "SortInfo.html");
rigthRoute("submit-b", ".some", "Binary.html", "BinaryInfo.html");
rigthRoute("submit-f", ".some3", "SortBubble.html", "SortBubbleInfo.html");
rigthRoute(
  "submit-s",
  ".some4",
  "SortSelection.html",
  "SortSelectionInfo.html"
);
rigthRoute("submit-a", ".some5", "SortInsert.html", "SortInsertInfo.html");
rigthRoute("submit-h", ".some6", "SortMerge.html", "SortMergeInfo.html");
rigthRoute("submit-j", ".some7", "SortQuick.html", "SortQuickInfo.html");

// тесты сортировки БЫСТРОЕ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortQuick-my-answer-code",
  "rigthM",
  "leftM",
  ["secondIndex", "j--", "items"]
);

// тесты сортировки ПУЗЫРЬКОМ
```

```
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(".code-input", "sort-my-answer-code", "rigthM", "leftM", [
  "endj",
  "j",
  "wasSwap",
]);

// тесты сортировки СЛИЯНИЕМ

rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortMerge-my-answer-code",
  "rigthM",
  "leftM",
  ["2", "middle", "resultArray"]
);

// тесты сортировки ВИБОРОМ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortSelection-my-answer-code",
  "rigthM",
  "leftM",
  ["arr[i]", "arr[min]", "tmp"]
);

// тестi сортировки ВСТАВКАМИ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortInsert-my-answer-code",
  "rigthM",
  "leftM",
  ["i - 1", "inputArr[j]", "[j + 1]"]
);

// тесты БИНАРНОГО ПОИСКА поиска
rightModal(
```

```
    "binary-my-answer-theory",
    "binary-in-answer",
    "rigthM",
    "leftM",
    7
  );
  inputTasksSubmit(".code-input", "binary-my-answer-code", "rigthM", "leftM",
[
    "mid",
    "mid",
    "mid",
  ]);

  // ответы на сортировке позырьком
  showRightAnswer("dang", "answerModal");
  showRightAnswer("dang-code", "code-task-right-a");

  // ответы бинарного поиска
  showRightAnswer("dang-code-binary", "code-task-right-a");
  showRightAnswer("dang-binary-theory", "answerModal");
 });
 window.addEventListener("load", function () {
  if (
    this.localStorage.getItem("name") &&
    this.localStorage.getItem("password")
  ) {
    this.document.location.href = "List.html";
  }

  const submitRegister = (name, password) => {
    if (
      this.localStorage.getItem("name") === null &&
      this.localStorage.getItem("password") === null
    ) {
      this.localStorage.setItem("name", name);
      this.localStorage.setItem("password", password);
    }
  };

  if (this.document.getElementById("submitReg")) {
```

```
    let btn = this.document.getElementById("submitReg");
    btn.addEventListener("click", () => {
      let name = this.document.getElementById("nameReg").value;
      let password = this.document.getElementById("passwordReg").value;
      submitRegister(name, password);
      this.document.location.href = "List.html";
    });
  }
});

window.addEventListener("load", function () {
  const toggleBinaryButton = () => {
    let btnSuccess = this.document.getElementById("binarySuccess");
    let btnDanger = this.document.getElementById("binaryDanger");
    if (
      this.localStorage.getItem("mid") &&
      this.localStorage.getItem("binary")
    ) {
      btnDanger.classList.toggle("d-none");
    } else {
      btnSuccess.classList.toggle("d-none");
    }
  };

  const toggleBubbleButton = () => {
    let btnSuccess = this.document.getElementById("bubbleSuccess");
    let btnDanger = this.document.getElementById("bubbleDanger");
    if (
      this.localStorage.getItem("wasSwap") &&
      this.localStorage.getItem("endj") &&
      this.localStorage.getItem("j")
    ) {
      btnDanger.classList.toggle("d-none");
    } else {
      btnSuccess.classList.toggle("d-none");
    }
  };

  const toggleSelectionButton = () => {
    let btnSuccess = this.document.getElementById("selectionSuccess");
```

```
  let btnDanger = this.document.getElementById("selectionDanger");
  if (
    this.localStorage.getItem("arr[i]") &&
    this.localStorage.getItem("arr[min]") &&
    this.localStorage.getItem("tmp")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleInsertButton = () => {
  let btnSuccess = this.document.getElementById("insertSuccess");
  let btnDanger = this.document.getElementById("insertDanger");
  if (
    this.localStorage.getItem("i - 1") &&
    this.localStorage.getItem("inputArr[j]") &&
    this.localStorage.getItem("[j + 1]")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleMergeButton = () => {
  let btnSuccess = this.document.getElementById("mergeSuccess");
  let btnDanger = this.document.getElementById("mergeDanger");
  if (
    this.localStorage.getItem("2") &&
    this.localStorage.getItem("middle") &&
    this.localStorage.getItem("resultArray")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};
```

```
  const toggleQuickButton = () => {
    let btnSuccess = this.document.getElementById("quickSuccess");
    let btnDanger = this.document.getElementById("quickDanger");
    if (
      this.localStorage.getItem("secondIndex") &&
      this.localStorage.getItem("j--") &&
      this.localStorage.getItem("items")
    ) {
      btnDanger.classList.toggle("d-none");
    } else {
      btnSuccess.classList.toggle("d-none");
    }
  };
  toggleQuickButton();
  toggleMergeButton();
  toggleInsertButton();
  toggleSelectionButton();
  toggleBubbleButton();
  toggleBinaryButton();
});

window.addEventListener("load", function () {
  if (
    !this.localStorage.getItem("name") &&
    !this.localStorage.getItem("password")
  ) {
    this.document.location.href = "register.html";
  }

  function arrSr(arr1, arr2) {
    if (arr1.length !== arr2.length) {
      return false;
    }
    for (let i = 0; i < arr1.length; i++) {
      if (arr1[i] !== arr2[i].getAttribute("data-val")) {
        return false;
      }
    }
    return true;
  }
```

```
let answerRightArr = ["a", "b", "c"];

function rigthRoute(btn, classN, strRoute1, strRoute2) {
  let mainBtn = this.document.getElementById(btn);
  arr = [];

  if (mainBtn) {
    mainBtn.addEventListener("click", () => {
      let answerArr = this.document.querySelectorAll(classN);

      if (answerArr.length > 0) {
        answerArr.forEach((item) => {
          if (item.checked) {
            arr.push(item);
          }
        });
      }

      if (arrSr(answerRightArr, arr)) {
        if (answerArr.length > 0) {
          this.document.location.href = strRoute1;
        }
      } else {
        this.document.location.href = strRoute2;
      }
      arr = [];
    });
  }
}

function rightModal(btn, input, rigth, left, rightAns) {
  let btnSubmitAns = this.document.getElementById(btn);
  let inputAns = this.document.getElementById(input);
  let rigthId = this.document.getElementById(rigth).getAttribute("id");
  let leftId = this.document.getElementById(left).getAttribute("id");
  let rightAnswer = rightAns;

  if (btnSubmitAns) {
    btnSubmitAns.addEventListener("click", () => {
```

```
      if (inputAns.value == rightAnswer) {
        this.localStorage.setItem("binary", rightAnswer);
        $("#" + rigthId).modal();
      } else {
        $("#" + leftId).modal();
      }
    });
  }
}

function showRightAnswer(btn, id) {
  let myBtn = this.document.getElementById(btn);
  let superId = this.document.getElementById(id).getAttribute("id");

  if (myBtn) {
    myBtn.addEventListener("click", () => {
      let confirnMass = this.prompt(
        "Ви упевнені що не в змозі вирішити задачу ? так | ні"
      );
      if (confirnMass == "так" || confirnMass == "т") {
        $("#" + superId).modal();
      } else if (confirnMass == "ні" || confirnMass == "н") {
        this.alert("Чудово намагайся далі !");
      }
    });
  }
}

function inputTasksSubmit(input, btn, rid, lid, arr) {
  let inputs = this.document.querySelectorAll(input);
  let codeBtn = this.document.getElementById(btn);
  let rightId = this.document.getElementById(rid).getAttribute("id");
  let leftId = this.document.getElementById(lid).getAttribute("id");
  let array = arr;

  if (codeBtn) {
    codeBtn.addEventListener("click", () => {
      let inputsArr = [];
      for (let i = 0; i < inputs.length; i++) {
        if (inputs[i].value !== "" && inputs[i].value !== null) {
```

```
        inputsArr.push(inputs[i].value);
       }
      }
      if (inputsArr.length !== array.length) {
       $("#" + leftId).modal();
       return;
      } else {
       if (
        inputsArr[0] == array[0] &&
        inputsArr[1] == array[1] &&
        inputsArr[2] == array[2]
       ) {
        this.localStorage.setItem(array[0], array[0]);
        this.localStorage.setItem(array[1], array[1]);
        this.localStorage.setItem(array[2], array[2]);
        $("#" + rightId).modal();
       } else {
        $("#" + leftId).modal();
       }
      }
    });
  }
}

// редиректы
//   rigthRoute("submit-c", ".some2", "Sort.html", "SortInfo.html");
rigthRoute("submit-b", ".some", "Binary.html", "BinaryInfo.html");
rigthRoute("submit-f", ".some3", "SortBubble.html", "SortBubbleInfo.html");
rigthRoute(
  "submit-s",
  ".some4",
  "SortSelection.html",
  "SortSelectionInfo.html"
);
rigthRoute("submit-a", ".some5", "SortInsert.html", "SortInsertInfo.html");
rigthRoute("submit-h", ".some6", "SortMerge.html", "SortMergeInfo.html");
rigthRoute("submit-j", ".some7", "SortQuick.html", "SortQuickInfo.html");

// тесты сортировки БЫСТРОЕ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
```

```
inputTasksSubmit(
  ".code-input",
  "sortQuick-my-answer-code",
  "rigthM",
  "leftM",
  ["secondIndex", "j--", "items"]
);

// тесты сортировки ПУЗЫРЬКОМ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(".code-input", "sort-my-answer-code", "rigthM", "leftM", [
  "endj",
  "j",
  "wasSwap",
]);

// тесты сортировки СЛИЯНИЕМ

rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortMerge-my-answer-code",
  "rigthM",
  "leftM",
  ["2", "middle", "resultArray"]
);

// тесты сортировки ВИБОРОМ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortSelection-my-answer-code",
  "rigthM",
  "leftM",
  ["arr[i]", "arr[min]", "tmp"]
);

// тестi сортировки ВСТАВКАМИ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
```

```
      ".code-input",
      "sortInsert-my-answer-code",
      "rigthM",
      "leftM",
      ["i - 1", "inputArr[j]", "[j + 1]"]
    );

    // тесты БИНАРНОГО ПОИСКА поиска
    rightModal(
      "binary-my-answer-theory",
      "binary-in-answer",
      "rigthM",
      "leftM",
      7
    );
    inputTasksSubmit(".code-input", "binary-my-answer-code", "rigthM", "leftM",
[
      "mid",
      "mid",
      "mid",
    ]);

    // ответы на сортировке позырьком
    showRightAnswer("dang", "answerModal");
    showRightAnswer("dang-code", "code-task-right-a");

    // ответы бинарного поиска
    showRightAnswer("dang-code-binary", "code-task-right-a");
    showRightAnswer("dang-binary-theory", "answerModal");
  });
  window.addEventListener("load", function () {
    if (
      this.localStorage.getItem("name") &&
      this.localStorage.getItem("password")
    ) {
      this.document.location.href = "List.html";
    }

    const submitRegister = (name, password) => {
      if (
```

```
      this.localStorage.getItem("name") === null &&
      this.localStorage.getItem("password") === null
    ) {
      this.localStorage.setItem("name", name);
      this.localStorage.setItem("password", password);
    }
  };

  if (this.document.getElementById("submitReg")) {
    let btn = this.document.getElementById("submitReg");
    btn.addEventListener("click", () => {
      let name = this.document.getElementById("nameReg").value;
      let password = this.document.getElementById("passwordReg").value;
      submitRegister(name, password);
      this.document.location.href = "List.html";
    });
  }
});


window.addEventListener("load", function () {
  const toggleBinaryButton = () => {
    let btnSuccess = this.document.getElementById("binarySuccess");
    let btnDanger = this.document.getElementById("binaryDanger");
    if (
      this.localStorage.getItem("mid") &&
      this.localStorage.getItem("binary")
    ) {
      btnDanger.classList.toggle("d-none");
    } else {
      btnSuccess.classList.toggle("d-none");
    }
  };

  const toggleBubbleButton = () => {
    let btnSuccess = this.document.getElementById("bubbleSuccess");
    let btnDanger = this.document.getElementById("bubbleDanger");
    if (
      this.localStorage.getItem("wasSwap") &&
      this.localStorage.getItem("endj") &&
```

```
    this.localStorage.getItem("j")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleSelectionButton = () => {
  let btnSuccess = this.document.getElementById("selectionSuccess");
  let btnDanger = this.document.getElementById("selectionDanger");
  if (
    this.localStorage.getItem("arr[i]") &&
    this.localStorage.getItem("arr[min]") &&
    this.localStorage.getItem("tmp")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleInsertButton = () => {
  let btnSuccess = this.document.getElementById("insertSuccess");
  let btnDanger = this.document.getElementById("insertDanger");
  if (
    this.localStorage.getItem("i - 1") &&
    this.localStorage.getItem("inputArr[j]") &&
    this.localStorage.getItem("[j + 1]")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleMergeButton = () => {
  let btnSuccess = this.document.getElementById("mergeSuccess");
  let btnDanger = this.document.getElementById("mergeDanger");
  if (
```

```
      this.localStorage.getItem("2") &&
      this.localStorage.getItem("middle") &&
      this.localStorage.getItem("resultArray")
    ) {
      btnDanger.classList.toggle("d-none");
    } else {
      btnSuccess.classList.toggle("d-none");
    }
  };

  const toggleQuickButton = () => {
    let btnSuccess = this.document.getElementById("quickSuccess");
    let btnDanger = this.document.getElementById("quickDanger");
    if (
      this.localStorage.getItem("secondIndex") &&
      this.localStorage.getItem("j--") &&
      this.localStorage.getItem("items")
    ) {
      btnDanger.classList.toggle("d-none");
    } else {
      btnSuccess.classList.toggle("d-none");
    }
  };
  toggleQuickButton();
  toggleMergeButton();
  toggleInsertButton();
  toggleSelectionButton();
  toggleBubbleButton();
  toggleBinaryButton();
});

window.addEventListener("load", function () {
  if (
    !this.localStorage.getItem("name") &&
    !this.localStorage.getItem("password")
  ) {
    this.document.location.href = "register.html";
  }

  function arrSr(arr1, arr2) {
```

```
  if (arr1.length !== arr2.length) {
    return false;
  }
  for (let i = 0; i < arr1.length; i++) {
    if (arr1[i] !== arr2[i].getAttribute("data-val")) {
      return false;
    }
  }
  return true;
}

let answerRightArr = ["a", "b", "c"];

function rigthRoute(btn, classN, strRoute1, strRoute2) {
  let mainBtn = this.document.getElementById(btn);
  arr = [];

  if (mainBtn) {
    mainBtn.addEventListener("click", () => {
      let answerArr = this.document.querySelectorAll(classN);

      if (answerArr.length > 0) {
        answerArr.forEach((item) => {
          if (item.checked) {
            arr.push(item);
          }
        });
      }

      if (arrSr(answerRightArr, arr)) {
        if (answerArr.length > 0) {
          this.document.location.href = strRoute1;
        }
      } else {
        this.document.location.href = strRoute2;
      }
      arr = [];
    });
  }
}
```

```
function rightModal(btn, input, rigth, left, rightAns) {
  let btnSubmitAns = this.document.getElementById(btn);
  let inputAns = this.document.getElementById(input);
  let rigthId = this.document.getElementById(rigth).getAttribute("id");
  let leftId = this.document.getElementById(left).getAttribute("id");
  let rightAnswer = rightAns;

  if (btnSubmitAns) {
    btnSubmitAns.addEventListener("click", () => {
      if (inputAns.value == rightAnswer) {
        this.localStorage.setItem("binary", rightAnswer);
        $("#" + rigthId).modal();
      } else {
        $("#" + leftId).modal();
      }
    });
  }
}

function showRightAnswer(btn, id) {
  let myBtn = this.document.getElementById(btn);
  let superId = this.document.getElementById(id).getAttribute("id");

  if (myBtn) {
    myBtn.addEventListener("click", () => {
      let confirnMass = this.prompt(
        "Ви упевнені що не в змозі вирішити задачу ? так | ні"
      );
      if (confirnMass == "так" || confirnMass == "т") {
        $("#" + superId).modal();
      } else if (confirnMass == "ні" || confirnMass == "н") {
        this.alert("Чудово намагайся далі !");
      }
    });
  }
}

function inputTasksSubmit(input, btn, rid, lid, arr) {
  let inputs = this.document.querySelectorAll(input);
```

```
  let codeBtn = this.document.getElementById(btn);
  let rightId = this.document.getElementById(rid).getAttribute("id");
  let leftId = this.document.getElementById(lid).getAttribute("id");
  let array = arr;

  if (codeBtn) {
    codeBtn.addEventListener("click", () => {
      let inputsArr = [];
      for (let i = 0; i < inputs.length; i++) {
        if (inputs[i].value !== "" && inputs[i].value !== null) {
          inputsArr.push(inputs[i].value);
        }
      }
      if (inputsArr.length !== array.length) {
        $("#" + leftId).modal();
        return;
      } else {
        if (
          inputsArr[0] == array[0] &&
          inputsArr[1] == array[1] &&
          inputsArr[2] == array[2]
        ) {
          this.localStorage.setItem(array[0], array[0]);
          this.localStorage.setItem(array[1], array[1]);
          this.localStorage.setItem(array[2], array[2]);
          $("#" + rightId).modal();
        } else {
          $("#" + leftId).modal();
        }
      }
    });
  }
}

// редиректы
//   rigthRoute("submit-c", ".some2", "Sort.html", "SortInfo.html");
rigthRoute("submit-b", ".some", "Binary.html", "BinaryInfo.html");
rigthRoute("submit-f", ".some3", "SortBubble.html", "SortBubbleInfo.html");
rigthRoute(
  "submit-s",
```

```
  ".some4",
  "SortSelection.html",
  "SortSelectionInfo.html"
);
rigthRoute("submit-a", ".some5", "SortInsert.html", "SortInsertInfo.html");
rigthRoute("submit-h", ".some6", "SortMerge.html", "SortMergeInfo.html");
rigthRoute("submit-j", ".some7", "SortQuick.html", "SortQuickInfo.html");

// тесты сортировки БЫСТРОЕ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortQuick-my-answer-code",
  "rigthM",
  "leftM",
  ["secondIndex", "j--", "items"]
);

// тесты сортировки ПУЗЫРЬКОМ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(".code-input", "sort-my-answer-code", "rigthM", "leftM", [
  "endj",
  "j",
  "wasSwap",
]);

// тесты сортировки СЛИЯНИЕМ

rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
  ".code-input",
  "sortMerge-my-answer-code",
  "rigthM",
  "leftM",
  ["2", "middle", "resultArray"]
);

// тесты сортировки ВИБОРОМ
rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
inputTasksSubmit(
```

```
    ".code-input",
    "sortSelection-my-answer-code",
    "rigthM",
    "leftM",
    ["arr[i]", "arr[min]", "tmp"]
  );

  // тестi сортировки ВСТАВКАМИ
  rightModal("binary-my-answer", "binary-in-answer", "rigthM", "leftM", 7);
  inputTasksSubmit(
    ".code-input",
    "sortInsert-my-answer-code",
    "rigthM",
    "leftM",
    ["i - 1", "inputArr[j]", "[j + 1]"]
  );

  // тесты БИНАРНОГО ПОИСКА поиска
  rightModal(
    "binary-my-answer-theory",
    "binary-in-answer",
    "rigthM",
    "leftM",
    7
  );
  inputTasksSubmit(".code-input", "binary-my-answer-code", "rigthM", "leftM",
[
    "mid",
    "mid",
    "mid",
  ]);

  // ответы на сортировке позырьком
  showRightAnswer("dang", "answerModal");
  showRightAnswer("dang-code", "code-task-right-a");

  // ответы бинарного поиска
  showRightAnswer("dang-code-binary", "code-task-right-a");
  showRightAnswer("dang-binary-theory", "answerModal");
});
```

```
window.addEventListener("load", function () {
  if (
    this.localStorage.getItem("name") &&
    this.localStorage.getItem("password")
  ) {
    this.document.location.href = "List.html";
  }

  const submitRegister = (name, password) => {
    if (
      this.localStorage.getItem("name") === null &&
      this.localStorage.getItem("password") === null
    ) {
      this.localStorage.setItem("name", name);
      this.localStorage.setItem("password", password);
    }
  };

  if (this.document.getElementById("submitReg")) {
    let btn = this.document.getElementById("submitReg");
    btn.addEventListener("click", () => {
      let name = this.document.getElementById("nameReg").value;
      let password = this.document.getElementById("passwordReg").value;
      submitRegister(name, password);
      this.document.location.href = "List.html";
    });
  }
});



window.addEventListener("load", function () {
  const toggleBinaryButton = () => {
    let btnSuccess = this.document.getElementById("binarySuccess");
    let btnDanger = this.document.getElementById("binaryDanger");
    if (
      this.localStorage.getItem("mid") &&
      this.localStorage.getItem("binary")
    ) {
      btnDanger.classList.toggle("d-none");
    } else {
```

```
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleBubbleButton = () => {
  let btnSuccess = this.document.getElementById("bubbleSuccess");
  let btnDanger = this.document.getElementById("bubbleDanger");
  if (
    this.localStorage.getItem("wasSwap") &&
    this.localStorage.getItem("endj") &&
    this.localStorage.getItem("j")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleSelectionButton = () => {
  let btnSuccess = this.document.getElementById("selectionSuccess");
  let btnDanger = this.document.getElementById("selectionDanger");
  if (
    this.localStorage.getItem("arr[i]") &&
    this.localStorage.getItem("arr[min]") &&
    this.localStorage.getItem("tmp")
  ) {
    btnDanger.classList.toggle("d-none");
  } else {
    btnSuccess.classList.toggle("d-none");
  }
};

const toggleInsertButton = () => {
  let btnSuccess = this.document.getElementById("insertSuccess");
  let btnDanger = this.document.getElementById("insertDanger");
  if (
    this.localStorage.getItem("i - 1") &&
    this.localStorage.getItem("inputArr[j]") &&
    this.localStorage.getItem("[j + 1]")
  ) {
```

```
    btnDanger.classList.toggle("d-none");
   } else {
    btnSuccess.classList.toggle("d-none");
   }
  };

  const toggleMergeButton = () => {
   let btnSuccess = this.document.getElementById("mergeSuccess");
   let btnDanger = this.document.getElementById("mergeDanger");
   if (
    this.localStorage.getItem("2") &&
    this.localStorage.getItem("middle") &&
    this.localStorage.getItem("resultArray")
   ) {
    btnDanger.classList.toggle("d-none");
   } else {
    btnSuccess.classList.toggle("d-none");
   }
  };

  const toggleQuickButton = () => {
   let btnSuccess = this.document.getElementById("quickSuccess");
   let btnDanger = this.document.getElementById("quickDanger");
   if (
    this.localStorage.getItem("secondIndex") &&
    this.localStorage.getItem("j--") &&
    this.localStorage.getItem("items")
   ) {
    btnDanger.classList.toggle("d-none");
   } else {
    btnSuccess.classList.toggle("d-none");
   }
  };
  toggleQuickButton();
  toggleMergeButton();
  toggleInsertButton();
  toggleSelectionButton();
  toggleBubbleButton();
  toggleBinaryButton();
});
```

```
window.addEventListener("load", function () {
 if (
  !this.localStorage.getItem("name") &&
  !this.localStorage.getItem("password")
 ) {
  this.document.location.href = "register.html";
 }

 function arrSr(arr1, arr2) {
  if (arr1.length !== arr2.length) {
   return false;
  }
  for (let i = 0; i < arr1.length; i++) {
   if (arr1[i] !== arr2[i].getAttribute("data-val")) {
    return false;
   }
  }
  return true;
 }

 let answerRightArr = ["a", "b", "c"];

 function rigthRoute(btn, classN, strRoute1, strRoute2) {
  let mainBtn = this.document.getElementById(btn);
  arr = [];

  if (mainBtn) {
   mainBtn.addEventListener("click", () => {
    let answerArr = this.document.querySelectorAll(classN);

    if (answerArr.length > 0) {
     answerArr.forEach((item) => {
      if (item.checked) {
       arr.push(item);
      }
     });
    }

    if (arrSr(answerRightArr, arr)) {
```

```
        if (answerArr.length > 0) {
          this.document.location.href = strRoute1;
        }
      } else {
        this.document.location.href = strRoute2;
      }
      arr = [];
    });
  }
}

function rightModal(btn, input, rigth, left, rightAns) {
  let btnSubmitAns = this.document.getElementById(btn);
  let inputAns = this.document.getElementById(input);
  let rigthId = this.document.getElementById(rigth).getAttribute("id");
  let leftId = this.document.getElementById(left).getAttribute("id");
  let rightAnswer = rightAns;

  if (btnSubmitAns) {
    btnSubmitAns.addEventListener("click", () => {
      if (inputAns.value == rightAnswer) {
        this.localStorage.setItem("binary", rightAnswer);
        $("#" + rigthId).modal();
      } else {
        $("#" + leftId).modal();
      }
    });
  }
}
```