ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ «ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ДЕННОЇ ОСВІТИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Допускається до захисту
Завідувач кафедри _____О.В. Ольховська
(підпис)
«_____»_____2022 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО ДИПЛОМНОЇ РОБОТИ

на тему:

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТРЕНАЖЕРУ З ТЕМИ «ВЕРИФІКАЦІЯ ПРОГРАМ» АНГЛОМОВНОГО ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи: Сімперович Микита Михайлович
_____«__»_____2022р.
(підпис)
Науковий керівник: к.ф.-м.н., доц. Черненко Оксана Олексіївна
_____ «__»_____2022р.
(підпис)

Полтава – 2022

ЗМІСТ

РЕФЕРАТ

Записка: 60 сторінок, 32 малюнка, 4 додатків, 23 джерела.

Предмет розробки: програмне забезпечення тренажеру з теми «Верифікація програм» англомовного дистанційного навчального курсу «Теорія програмування».

Мета роботи: розробка тренажеру з теми «Верифікація програм» англомовного дистанційного навчального курсу «Теорія програмування», що забезпечить закріплення знать з певної теми.

Методи, які були використані для розв'язання задачі: об'єктно-орієнтоване програмування, розробка інтерфейсу користувача виконана з використанням мови гіпертекстової розмітки HTML та таблиць стилів CSS, для реалізації функціоналу форми було використано мову програмування JavaScript, застосування системи контролю версій GitHub.

Ключові слова: тренажер, верифікація, форма, інтерфейс, специфікація, бібліотеки JavaScript, HTML, CSS, фреймворк, запит JSON, GitHub.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	Пояснення умовних позначень, символів, одиниць, скорочень, термінів
HTTP	Hyper Text Transfer Protocol
CSS	Cascading Style Sheets
NPM	Node Package Manager
JS	Java Script
Reakt	Бібліотека Java Script
Bootstrap	Набір інструментів для створення сайтів
Node.js	Платформа для виконання Java Script коду
ASP.NET	Active Server Pages для .NET
GitHub	Система управління проектами і версіями
ПЗ	програмне забезпечення
ООП	об’єктно - орієнтоване програмування

ВСТУП

Дистанційна форма навчання це сучасна форма надання освітніх послуг, рівноцінна класичній університетській освіті, що здійснюється за допомогою засобів Інтернет. Дистанційне навчання ставить на меті не тільки навчання в своїй традиційній формі, але ще передбачає напрацювання у студента навичок самостійної роботи.

Успішність дистанційного навчання залежить від ефективної її організації, від керівництва самим процесом і майстерності викладачів, що приймають в ньому участь. Вивчення інформаційних та математичних дисциплін у випадку дистанційного навчання має свою специфіку і особливості, які дозволяють значно розширити види навчальної роботи порівняно з традиційним навчанням.

Однією із можливостей підвищення якості освітнього процесу в ВНЗ є використання навчальних програм-тренажерів. Їх використання дозволяє значно скоротити час вивчення і закріплення нового матеріалу, чим прискорити сам процес навчання. При виконанні завдань студенти можуть наглядно розібрати задачі та алгоритми стільки разів, скільки їм потрібно для розуміння матеріалу, при цьому тренажер може використовуватися як на заняттях так і самостійно. Самостійне вирішення задач сприяє розвитку мислення та закріпленню необхідних навичок. Тренажер, повинен наглядно демонструвати процес роботи досліджуваних задач та алгоритмів.

У зв'язку з тим, що кількість іноземних студентів в українських вузах з кожним роком зростає, зараз важливим є питання розробки англомовних курсів навчання для іноземних студентів. Раніше для іноземних студентів викладання проводилося тільки українською мовою, рідше – російською. Через це відбувалося безліч непорозумінь і конфузів, тому що більшість іноземних студентів, які вступають до вузів України недостатньо володіють українською мовою.

Тому метою даної роботи є розробка програмного забезпечення (ПЗ) тренажеру з теми «Верифікація програм» для англомовного дистанційного навчального курсу «Теорія програмування».

Важливим моментом є можливість розміщення тренажерів на веб-сторінках. Дана технологія дозволяє забезпечити загальнодоступність створюваних засобів.

Об’єктом розробки в даній роботі є процес дистанційного навчання з дисципліни «Теорія програмування».

Предметом розробки є програмний продукт, що реалізує тренажер для закріплення знань з теми «Верифікація програм».

Головне завдання – розробити алгоритм роботи тренажера, по якому потрібно створити навчальний тренажер з теми «Верифікація програм», як складову дистанційного курсу з дисципліни «Теорія програмування».

Робота складається з постановки задачі , інформаційного огляду, теоретичної та практичної частин. У розділі «Постановка задачі» розглянуті основні кроки при реалізації проекту та основні вимог до ПЗ, інформаційний огляд включає аналіз інших подібних навчальних тренажерів, у теоретичній частині надано теоретичний матеріал з теми «Верифікація програм», алгоритм роботи тренажеру, у практичній частині описано програмні засоби, які були використані при реалізації проекту, процес програмної реалізації, опис роботи самої програми.

1. ПОСТАНОВКА ЗАДАЧІ

Задача магістерської дипломної роботи полягає у розробці алгоритму та реалізації програмного забезпечення тренажеру для навчального курсу «Теорія програмування» за темою «Верифікація програм», який повинен бути адаптовано для англомовних студентів.

Для виконання поставленого завдання необхідно:

- Знайти та проаналізувати навчальні тренажери з курсу «Теорія програмування» або з інших подібних курсів, наприклад «Теорія алгоритмів» , «Теорія ймовірностей та математична статистика», «Елементи комбінаторної оптимізації», «Теорія алгоритмів», «Інформатика», «Алгоритми та структури даних», «Математична логіка» та інші.
- Ознайомитись та проаналізувати теоретичний матеріал за темою «Верифікація програм».
- Розробити завдання для тренажеру на англійській мові.
- Розробити алгоритм реалізації цих завдань.
- Розробити блок сему реалізації ПЗ навчального тренажеру
- Проаналізувати та обрати мову і середовище для створення ПЗ
- Створити ПЗ тренажеру за темою «Верифікація програм»
- Виконати тестування, та корегування ПЗ

На головному екрані тренажеру повинна міститись інформація про розробника та керівника проекту, а також назва теми тренажера та назва курсу.

Тренажер повинен містити декілька вправ з завданнями, після правильної відповіді на завдання користувач повинен перейти на наступний крок, якщо відповідь не вірна інтерфейс повинен видати підказку користувачеві.

Основні вимоги до ПЗ:

- Зручний та зрозумілий для користувача інтерфейс.
- ПЗ повинно бути реалізоване на англійській мові, так як навчальний тренажер призначено для англомовних студентів.

- На кожному кроці повинно показати умову задачі та варіанти відповідей.
- Після кожної відповіді повинна виконуватись перевірка на правильність, при невірній відповіді необхідно надати підказку студенту.
- На кожному кроці повинен бути доступ до інструкції користувача.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Інформаційні технології навчання

Інформаційна технологія навчання – це процес підготовки та передачі інформації студенту, засобом здійснення якого є комп'ютерна техніка (технічні засоби) та програмні засоби.

Один із найважливіших напрямів використання інформаційних технологій – застосування тренажерів – дуже важлива складова частина навчального процесу. Ці програми дозволяють відпрацювати конкретні знання, вміння, навички. Досвід застосування навчальних тренажерів дозволяє виділити такі позитивні моменти:

1) враховується індивідуальний темп роботи студента, який сам управляє навчальним процесом;

2) скорочується час вироблення необхідних навичок;

3) збільшується кількість тренувальних завдань;

4) легко досягається рівнева диференціація;

5) підвищується мотивація навчальної діяльності. [9]

Інший напрямок – застосування програм-тестів. На етапі навчання застосування тестів є необхідним компонентом навчання. У умовах використання програм-тестів є дуже актуальним. Велика рутинна робота, пов'язана з перевіркою тестів та їх відпрацюванням, покладається на комп'ютер, що звільняє час викладача.

Важливим у роботі викладача є організація контролю за знаннями студентів. І тут використання інформаційних технологій відіграє важливу роль. Більшість електронних підручників містить вправи – тренажери, завдання з рішеннями, тестові завдання. Окремі програмні продукти містять електронний журнал, який дозволяє фіксувати рівень знань учня з кожної теми курсу (враховується як відмітка і кількість спроб розв'язання, так й витрачений час виконання завдань). Система оцінки результатів дає можливість визначити

рейтинг студента з кожної теми, простежити динаміку успішності та скоригувати навчальний процес відповідно до наведених результатів. Крім того, використання контролюючих програм сприяє формуванню адекватної самооцінки учнів. [9]

Ще один популярний напрямок впровадження інформаційних технологій в освітній процес – використання інтернет - тренажерів. Інтернет - тренажери – це програмно-методичний комплекс, в основу якого покладено методику критеріального оцінювання знань, умінь студентів, систему діагностики та інтерпретації отриманих відповідей, алгоритми цілеспрямованого тренування студентів у процесі багаторазового повторного виконання тестових завдань та пояснень причин їх невиконання.[9]

2.2 Огляд подібних до теми роботи тренажерів

Для огляду та аналізу подібних за темою роботи тренажерів було обрано тренажери з дистанційного курсу «Аналіз алгоритмів» за темами «1R - алгоритм», «Аналіз алгоритму сортування бульбашками» та «Аналіз алгоритму швидкого сортування»

Тренажер з теми «1R -алгоритм» розробив студент Мандрика Віталій, гр. КН-61 під керівник к.ф.-м.н. Олексійчук Ю.Ф. Після запуску програми для виконання завдання треба натиснути кнопку «Почати», після чого на екрані з’являється перше питання (рис.2.1), в відповідному полі треба ввести відповідь та натиснути на кнопку «Відповісти»

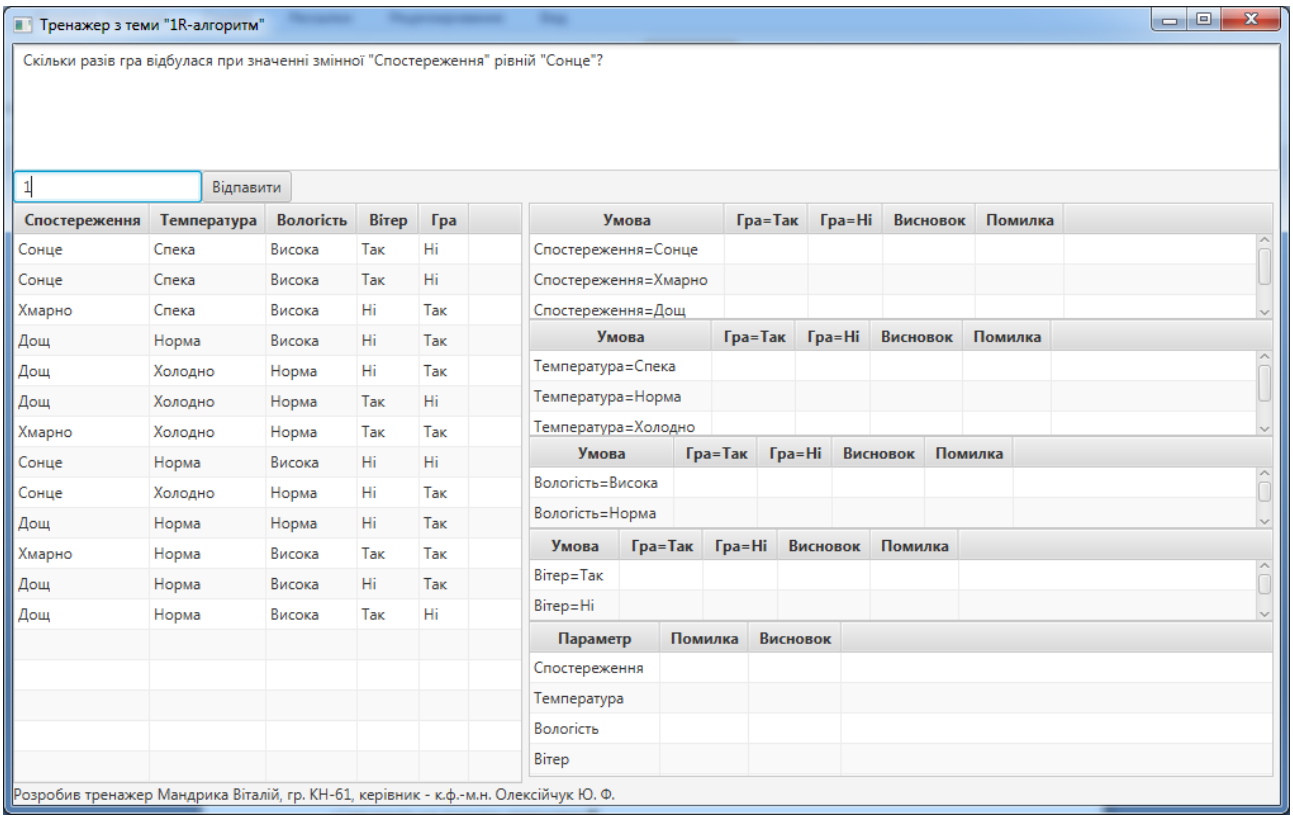

Рисунок 2.1 – Вікно тренажеру «1R - алгоритм»

Якщо відповідь вірна результат заноситься у таблицю відповідей, що зображено на рис 2.1.

Умова	Гра=Так	Гра=Ні	Висновок	Помилка
Спостереження=Сонце	1			
Спостереження=Хмарно				
Спостереження=Дощ				

Умова	Гра=Так	Гра=Ні	Висновок	Помилка
Температура=Спека				
Температура=Норма				
Температура=Холодно				

Умова	Гра=Так	Гра=Ні	Висновок	Помилка
Вологість=Висока				
Вологість=Норма				

Умова	Гра=Так	Гра=Ні	Висновок	Помилка
Вітер=Так				
Вітер=Ні				

Параметр	Помилка	Висновок
Спостереження		
Температура		
Вологість		
Вітер		

Рисунок 2.2 - Таблиця відповідей

Якщо відповідь не вірна на екрані з’являється повідомлення про помилку та пропозиція спробувати знову, як зображено на рис. 2.3.

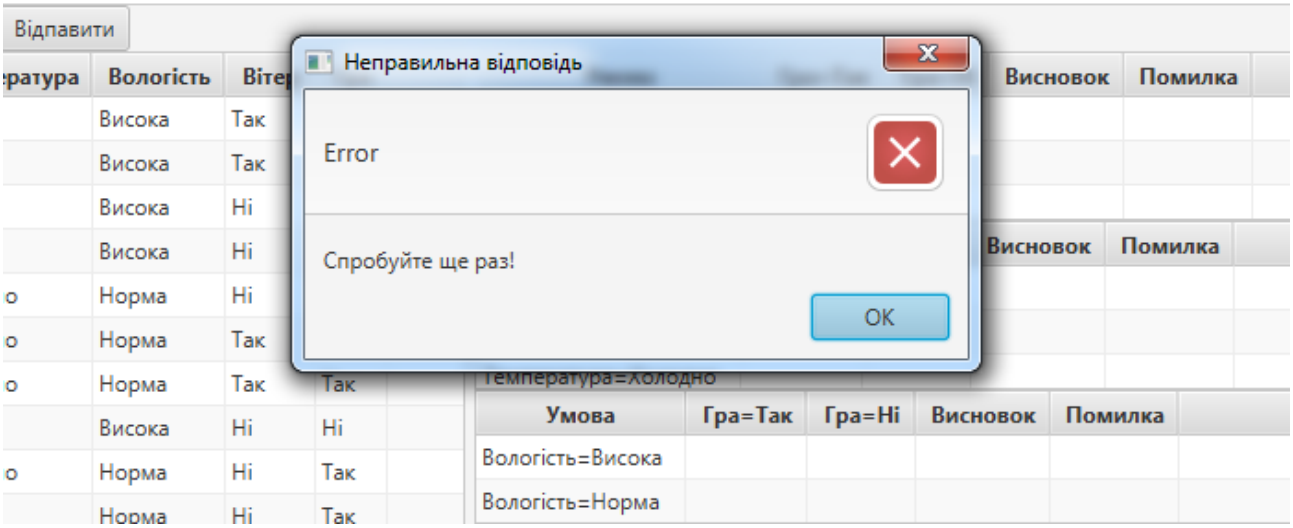

Рисунок 2.3 - Повідомлення про помилку

Якщо після трьох спроб відповідь не вірна, система видає підказку, як зображено на рис. 2.4.

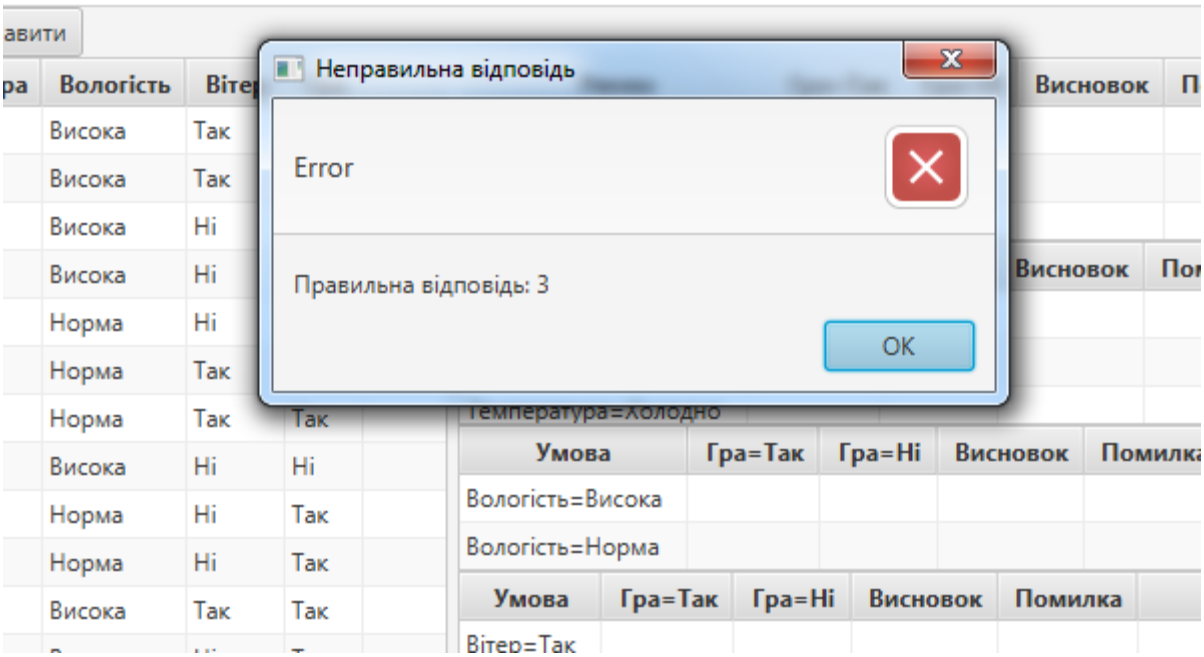

Рисунок 2.4 - Повідомлення з підказкою

Після виконання завдання система видає повідомлення, що робота завершена.

Головне вікно тренажеру «Аналіз сортування бульбашками» містить інформацію о задачі яку виконує даний тренажер, також є кнопка «Теоретичні відомості», при натисканні на яку відкривається теоретичний матеріал по темі алгоритму сортування бульбашками, що дуже зручно для користувача (рис. 2.5)

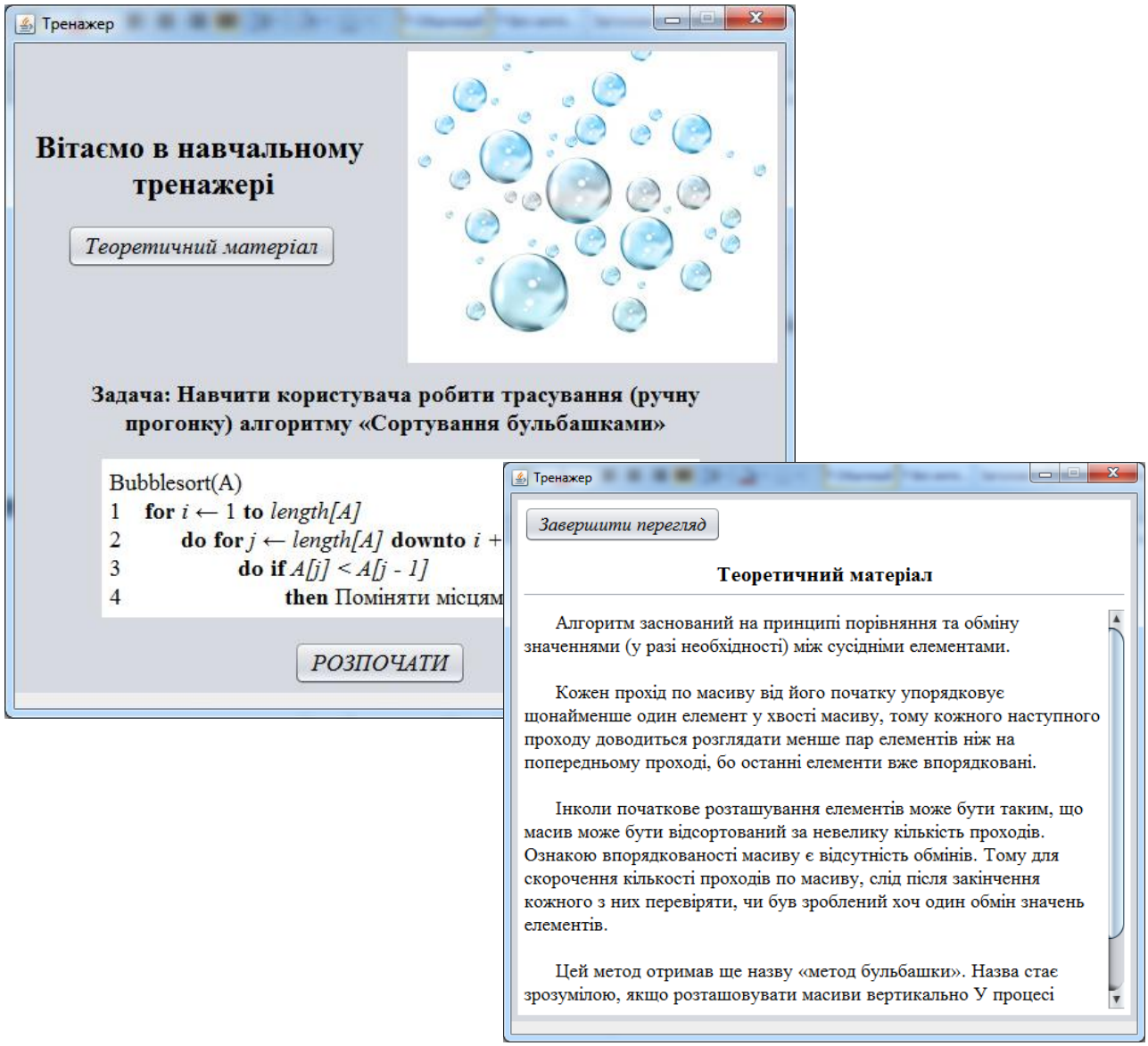

Рисунок 2.5 - Головне вікно тренажеру «Сортування бульбашками»

Після натискання на кнопку «Розпочати» відкривається вікно де відображено масив який треба відсортувати, кроки алгоритму сортування бульбашками та саме завдання зображено на рис.2.6.

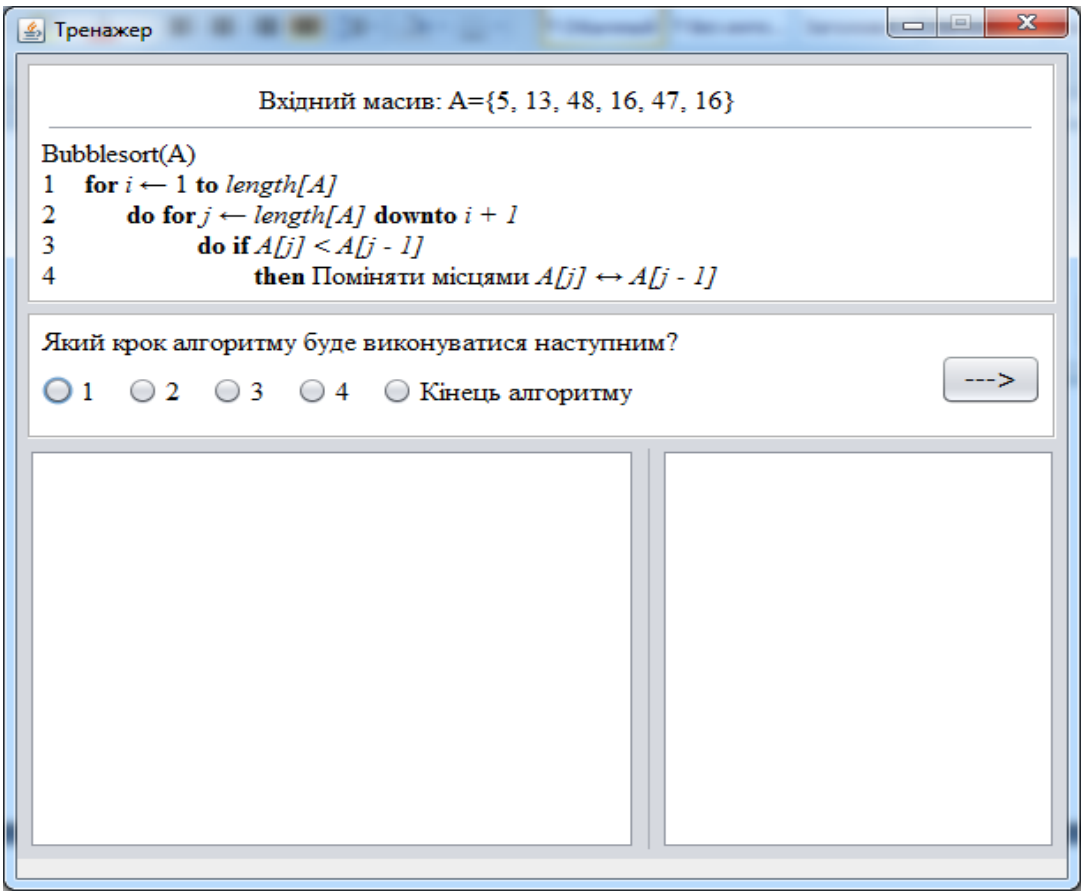

Рисунок 2.6 - Завдання тренажеру аналізу алгоритму «Сортування бульбашками»

Якщо відповідь вірна, то програма переходить до наступного кроку, якщо ж відповідь не вірна система видає повідомлення про помилку, як показано на рис. 2.7.

Рисунок 2.7 - Повідомлення про помилку

При повторному введені невірної відповіді, система видає підказку, як показано на рис. 2.8.

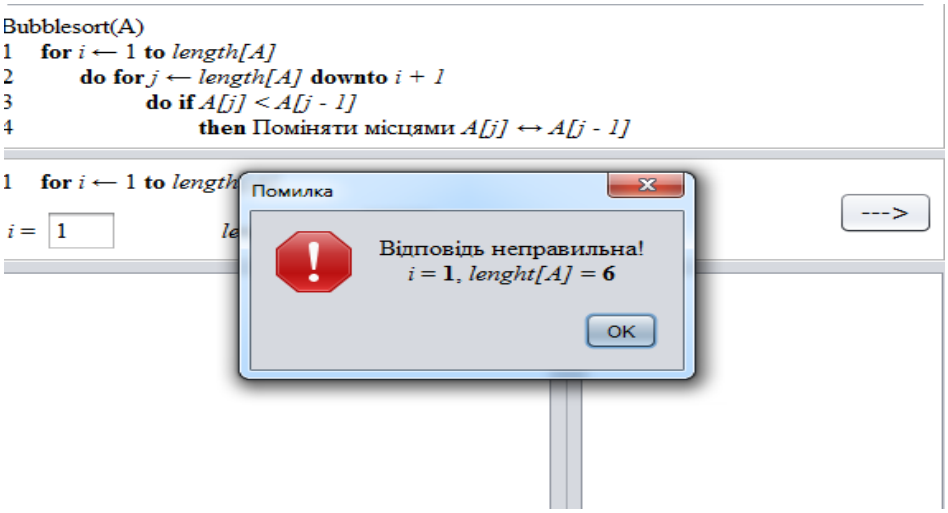

Рисунок 2.8 - Вікно з підказкою

Також в нижній частині екрана система відображає хід та результат виконання алгоритму, що є дуже зручно для користувача (рис. 2.9).

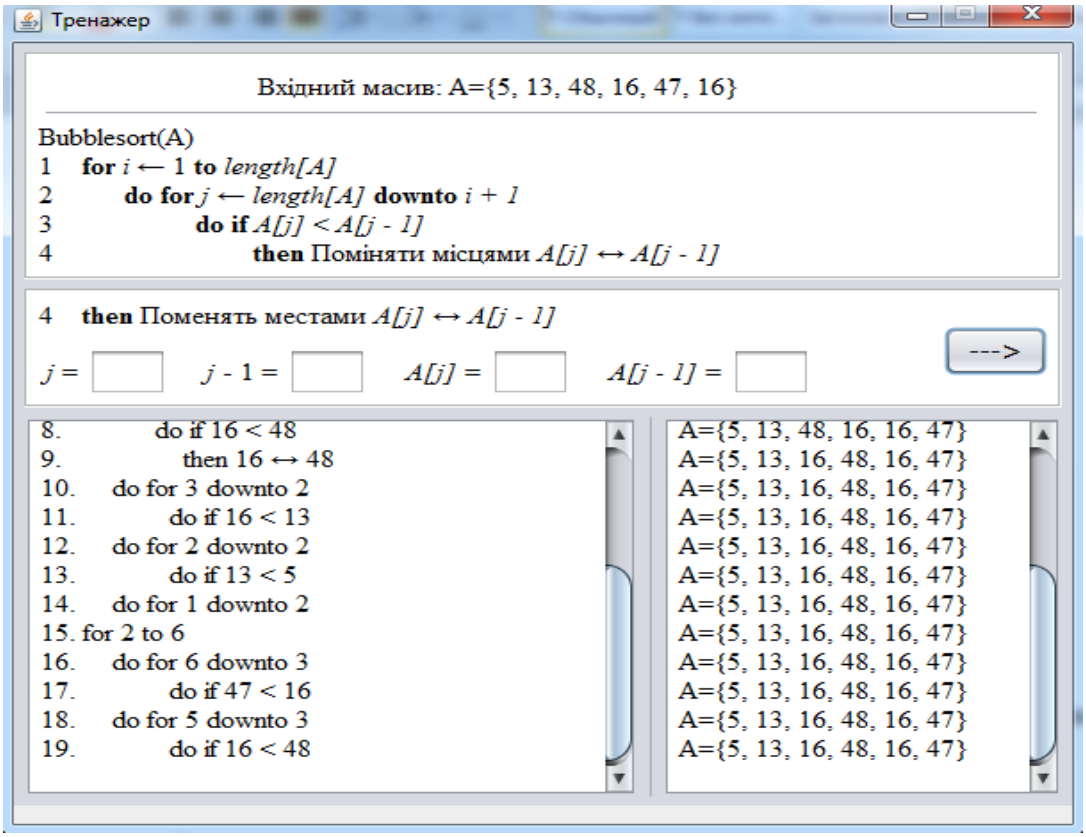

Рисунок 2.9 - Відображення результатів виконання алгоритму

Після завершення виконання завдання система пропонує виконати інший приклад.

Тренажер «Сортування бульбашками» пропонує покрокове проходження алгоритму сортування бульбашками, що сприяє більш доскональному розумінню студентом дію цього алгоритму.

Тренажер з теми «Аналіз алгоритму швидкого сортування» розробив студент Русін В.С, керівник Олексійчук Ю.Ф. На головному вікні програми відображається інформація про тему та дисципліну для якої було розроблене це програмне забезпечення, як зображено на рис. 2.10.

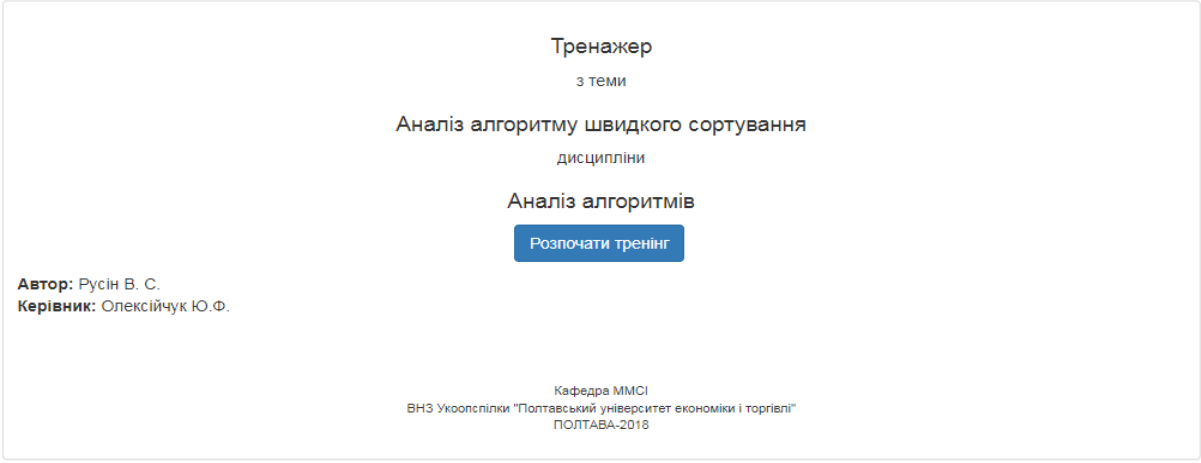

Рисунок 2.10 - Головна сторінка інтерфейсу тренажеру «Аналіз алгоритму швидкого сортування»

Після натискання на кнопку «Розпочати тренінг» відкривається сторінка де треба самостійно задати масив для сортування, як зображено на рис. 2.11

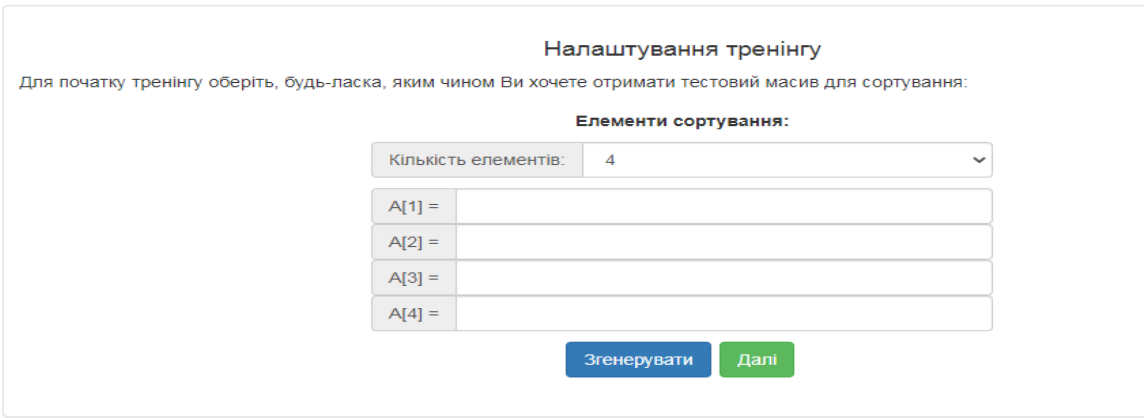

Рисунок 2.11 - Вікно для визначення масиву швидкого сортування

Після натискання на кнопку «Далі» на екрані відображається сторінка на якій знаходитися: функції алгоритму швидкого сортування, вхідні данні (елементи масиву та поточний стан масиву), блок де відображаються виконані дії та блок завдання де треба вибрати функцію алгоритму швидкого сортування, вказати крок виконання алгоритму у вибраній функції. Також на сторінці відображається кількість помилок при виконанні завдання, що зручно для самооцінки студента. (рис. 2.12).

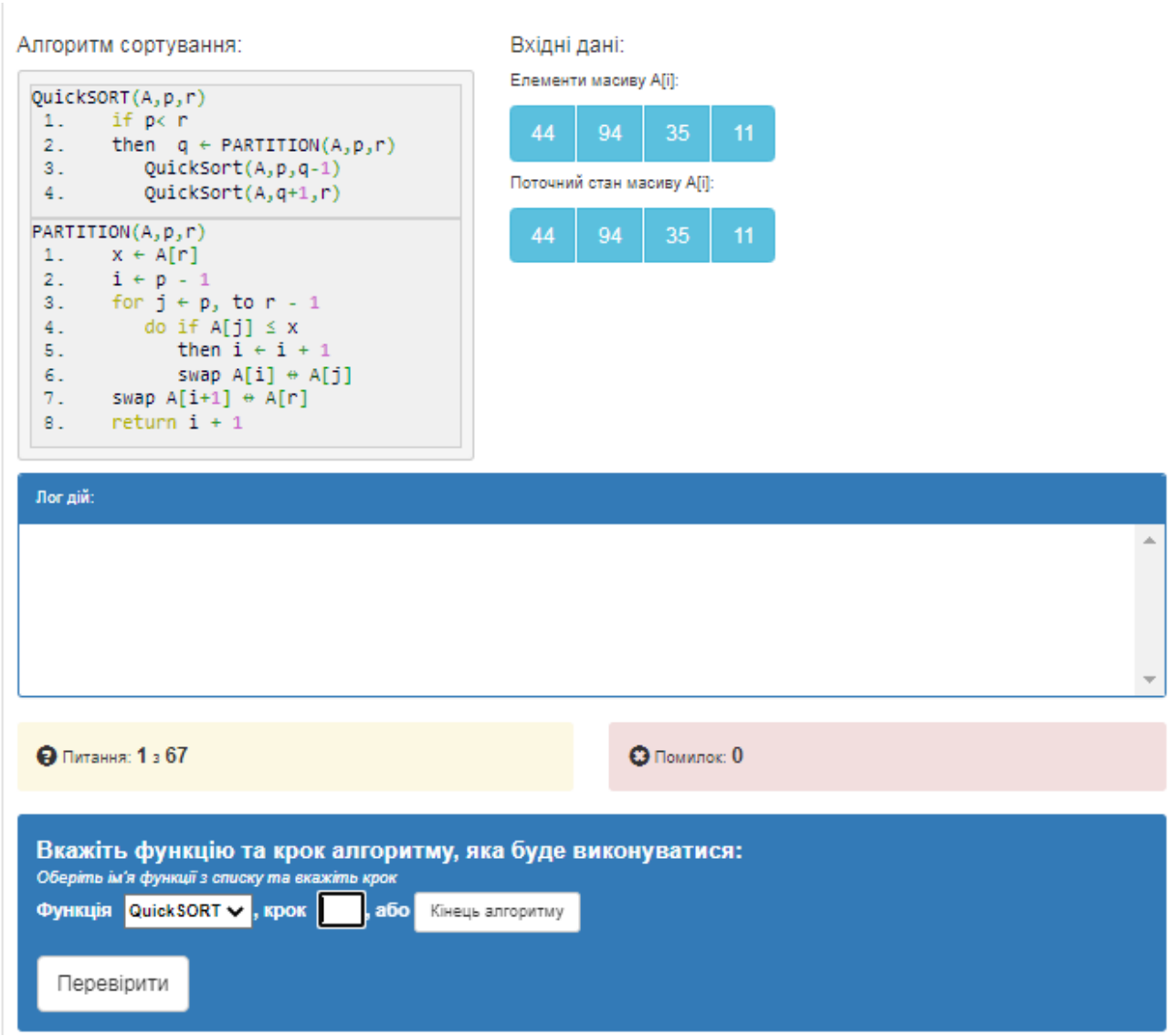

Рисунок 2.12 - Сторінка з покроковим завдання аналізу алгоритму швидкого сортування.

Якщо відповідь не вірна система відображає повідомлення про помилку та правильну відповідь, як показано рис. 2.13.

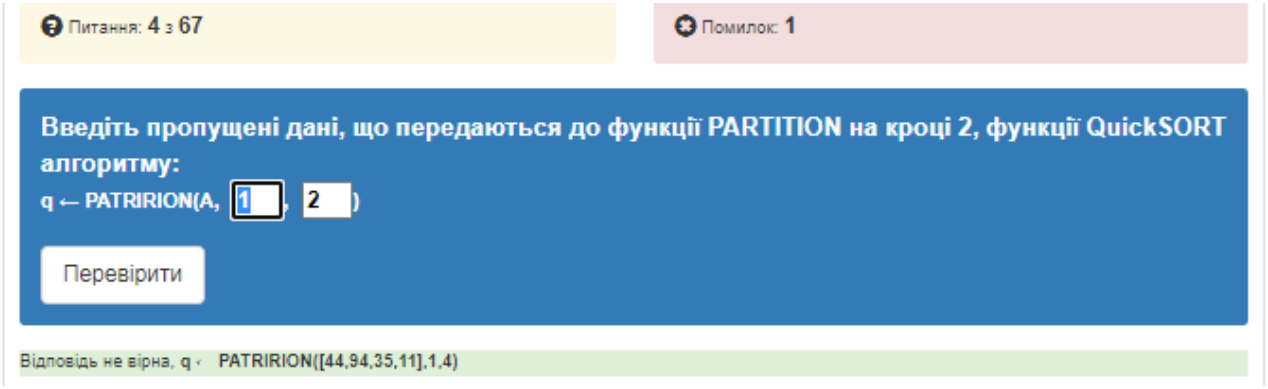

Рисунок 2.13 Блок з повідомлення про помилку та вірною відповіддю.

Після виконання завдання система пропонує почати знову та задати новий масив для виконання.

Тренажер з теми «Аналіз швидкого сортування» передбачає покрокове виконання програмного коду, що надає студенту більш глибокого розуміння роботи цього алгоритму.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Проблема верифікації програм. Загальні відомості

Проблеми верифікації (тобто докази правильності) програм займають центральне становище у теорії та практиці розробки програмного забезпечення. Під правильністю програм розуміється їх відповідність різним умовам коректності, безпеки, стійкості у разі непередбаченої поведінки оточення, ефективності використання ресурсів, часу та пам'яті, оптимальності реалізованих у програмі алгоритмів тощо. Як правило, для перевірки коректності роботи програми її тестують, тобто аналізують її поведінку на деяких вхідних даних. Проте тестування має очевидний недолік: якщо його можливо провести не для всіх допустимих вхідних даних, а тільки для невеликої частини (що має місце майже завжди), воно не може служити гарантованим обґрунтуванням того, що тестована програма володіє властивостями, що перевіряються. Як зазначив Дейкстра [3], тестування може лише допомогти виявити деякі помилки, але аж ніяк не довести їхню відсутність. Помилки в програмах можуть бути дуже тонкими, і що тонша помилка, тим складніше виявити її тестуванням. Але у багатьох програмах наявність навіть незначних помилок категорично неприпустима.

Наведемо приклад, що ілюструє наявність помилок навіть у дуже простих програм, правильність яких на перший погляд не викликає жодного сумніву. Розглянемо програму $\boldsymbol{P}$, завдання якої полягає у зарахуванні грошей на рахунок клієнта банку. Кількість грошей на рахунку цього клієнта зберігається у базі даних банку у змінній $\boldsymbol{P}$. Коли $\boldsymbol{P}$ виконує дії із зарахування суми **s** з цього приводу, вона виконує такі дії:

- копіює у свою внутрішню пам'ять значення змінної $\boldsymbol{x}$
- обчислює нове значення, яке має мати змінна $\boldsymbol{x}$, воно дорівнює сумі поточного значення $\boldsymbol{x}$ і сумі, що зараховується $\boldsymbol{s}$, і
- заносить до змінної $\boldsymbol{x}$ це нове значення.

Навіть якщо програма P виконує всі свої дії правильно, це не гарантує коректності обслуговування клієнта у разі, коли стан його рахунка може змінюватися кількома такими програмами. Розглянемо ситуацію, коли стан рахунку клієнта змінюють дві програми P_1 та P_2 описаного вище типу. Можливий наступний варіант спільного виконання цих програм:

- спочатку програма P_1 виконує свою першу дію, вона починається в момент часу t_1, а заключна дія P_1 (оновлення значення x) відбувається в момент часу t_2, та
- в момент часу, що лежить в інтервалі між t_1 та t_2, програма P_2 починає свою операцію зарахування грошей на рахунок клієнта, причому виконання першої дії програми P_2 (копіювання значення змінної x) проводиться до виконання заключної дії програми P_1.

Після завершення роботи обох програм ті гроші, які зарахувала на рахунок клієнта одна з програм P_1, P_2 просто пропадуть.

Помилку описаного вище типу не можливо знайти шляхом тестування, т.к. операція зарахування грошей на рахунок клієнта виконується практично миттєво, і тому серед тестів, якими можна аналізувати програми такого типу, з великою ймовірністю можуть відсутні такі тести, в яких дві різні програми, що обслуговують рахунки клієнтів, майже одночасно звертаються до одного та того ж ресурсу пам'яті.

Якщо ж в результаті якого-небудь тестування зазначена вище помилка виявляється, і для її виправлення конструюються спеціальні програмні механізми (семафори, локи, тощо) з метою завдання правильної дисципліни звернення програм до одного ресурсу пам'яті, то постає питання про те, наскільки ці механізми відповідають своєму призначенню (зокрема, чи захищені семафори від непередбаченої та неавторизованої зміни їх значень). Це також може аналізуватися шляхом тестування, і знову може вийти так, що серед тестів, якими аналізується поведінку зазначених вище спеціальних програмних механізмів, з великою ймовірністю будуть відсутні такі тести, в яких виявляється некоректна поведінка цих механізмів.

Гарантоване обґрунтування правильності програм може бути отримано лише за допомогою альтернативного підходу, принципово відмінного від тестування. Цей підхід називається **верифікацією**. У найзагальнішому вигляді верифікація програми може розумітися як побудова математичного доказу твердження про те, що програма, що верифікується, відповідає своєму призначенню. Призначення програми може бути виражене, наприклад, шляхом опису функції, яку повинна обчислювати ця програма, або правил взаємодії цієї програми з іншими програмами, тобто. реакції, яку ця програма повинна забезпечувати у відповідь отримання сигналів або повідомлень від інших програм. Формальний опис призначення програми (або деяких властивостей, яки вона повинна мати) у вигляді математичного твердження називається **специфікацією** цієї програми. Специфікація може являти собою формальний опис найрізноманітніших властивостей програми, наприклад:

- її вхідні та вихідні дані знаходяться у заданому співвідношенні, програма завжди завершує свою роботу,
- під час роботи програми не відбувається збоїв та ненормальних ситуацій (наприклад, поділу на 0, вилучення квадратного кореня з негативного числа, виходу індексу за межі масиву, неавторизованих витоків інформації, тощо),
- програма вирішує своє завдання за встановлений час,
- програма використовує трохи більше встановленого обсягу пам'яті.

Для перевірки програми P необхідно визначити:

- математичний сенс усіх конструкцій, що використовуються в P, називають **формальною семантикою** (або просто семантикою) цих конструкцій, та
- специфікацію $Spec$ цієї програми, що виражає ту властивість програми P, яку необхідно верифікувати,

після чого можна ставити питання про верифікацію P щодо $Spec$, тобто про побудову математичного доказу твердження про те, що P задовольняє $Spec$.[4]

У теорії програмування **семантика** – це інтерпретація (смислове значення) абстрактного синтаксису, виражена у термінах математично строгої формальної

моделі. Іншими словами, це інтерпретація множини припустимих конструкцій мови програмування.

Існує три основних підходи до формального опису семантики мов програмування.

Операційна семантика. Семантика описується в термінах деякої машини, реальної або віртуальної, наприклад, машини Тьюринга.

Смислове значення програми описується чітко описаною множиною операторів, які можуть бути реалізовані на цій машині. У результаті виконання операторів мови змінюється **стан програми** – сукупність значень її змінних. Результат виконання програми – це стан, при якому машина завершує роботу з даної програми, який, як правило, залежить від початкового стану.

Аксіоматична (деривaційна семантика). Семантику кожної синтаксичної конструкції мови задають у вигляді аксіом, яким повинні задовольняти вихідні дані, і правил виведення, за допомогою яких може бути отриманий результат (як теорема, що випливає з аксіом за допомогою правил виведення), тобто ця семантика є розширенням математичної теорії вирахування предикатів.

Метою роботи програми є досягнення завершального стану, що задовольняє якусь (бажану) умову. Цю умову називають **постумовою** і представляють предикатом Q.

Щоб коректно використовувати програму, потрібно знати початковий стан або множину початкових станів, при яких запуск програми обов'язково приведе до завершального стану, що задовольняє постумову Q. Умову, яку задовольняють ці початкові стани, називають **передумовою** і представляють предикатом P.

Нехай S – програма або оператор, або послідовність операторів.

Аксіоматична семантика оператора або деякого фрагмента програми задається у вигляді “трійки Хоара” {P}S{Q}. Запис {P}S{Q} означає: якщо виконання S почалося в стані, який задовольняє P, то є гарантія, що воно завершиться через скінченний час у стані, що задовольняє умову Q.

Умову, яка характеризує множину усіх станів, для яких виконання S, що почалося у такому стані, обов'язково закінчиться через скінченний час у стані, що задовольняє умову Q, називають найслабшою передумовою.

Для найслабшої передумови використовують позначення wp (S,Q) – weakest pre-condition.

Приклад 1

S: i:=i+1; Q: i <= 1;

wp (S, Q): i<=0 .

Найслабша умова означає найменш обмежуючу умову.

Наприклад: P1: (x >3) слабше, ніж P2: (x >5).

3.2 Індуктивне твердження Флойда, правила Хоара і верифікація

Визначемо спочатку поняття часткової і повної коректностіей програм.

Програма Prog **частково коректна** відносно передумови P і постумови Q, якщо для будь-якого початкового стану, де виконується P і для якого Prog завершує свою роботу, у завершальному стані буде виконуватися Q.

Програма Prog **повністю коректна** відносно передумови P і постумови Q, якщо для будь-якого початкового стану, де виконується P, правильно, що Prog завершує свою роботу і в завершальному стані буде виконуватися Q.

Один із методів верифікації програм, є метод індуктивних тверджень Флойда Розглянемо його на прикладі простої програми обчислення цілої частини квадратного кореня числа x. Блок-схема програми зображена на рис.3.1.

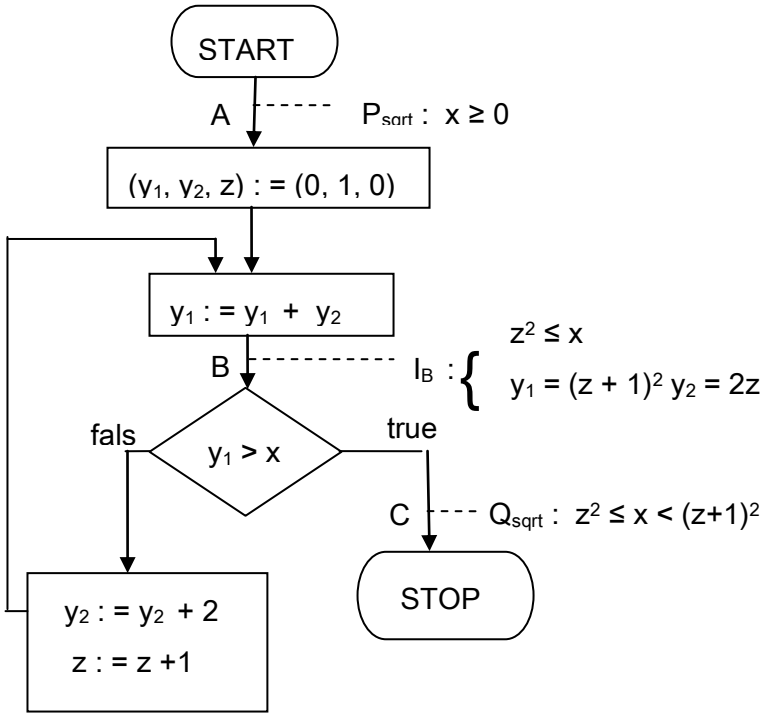

Рисунок 3.1 – Блок-схема програми

Схема доведення складається з трьох кроків.

Крок 1 – контрольні точки. Спочатку на ребрах блок-схеми вибирається множина контрольних точок. Ця множина повинна задовольняти такі умови.

1 Кожний цикл у блок-схемі містить хоча б одну контрольну точку.

2 Ребро, яке виходить з початкового оператора, має контрольну точку. Вона називається початковою точкою.

3 Для кожного завершального оператора ребро, яке веде до нього, має контрольну точку. Вона називається завершальною точкою.

На нашій блок-схемі такими точками є A, B і C.

Базовим шляхом називається такий шлях у блок-схемі, на кінцях якого знаходяться контрольні точки, і він не містить проміжних точок.

Для нашої блок-схеми базовими є такі шляхи:

1. Шлях α з точки A у точку B через оператори $(y_1, y_2, z) := (0, 1, 0)$ і $y_1 := y_1 + y_2$.
2. Шлях β з точки B у точку B через оператори $y_1 > x$, $y_2 := y_2 + 2$, $z := z + 1$ і $y_1 := y_1 + y_2$.

3. Шлях γ з точки B у точку C через оператор $y_1>x$.

Крок 2 – індуктивні твердження. На цьому кроці для кожної контрольної точки S вибирається формула I_S . Вона називається індуктивним твердженням для S. Індуктивне твердження I_S повинне бути інваріантом в S для передумови P. На наступному третьому кроці методу Флойда цей факт доводиться для всіх індуктивних тверджень.

Індуктивним твердженням для початкової точки повинна бути передумова P, а для завершальних – постумова Q. Таким чином, нарешті буде доведено, що Q – інваріант у кожній завершальній точці. З того, що Q - інваріант у кожній завершальній точці для передумови P, випливає часткова коректність програми.

Для даної блок-схеми запропонуємо такі індуктивні твердження:

$$I_A: \ P_{sqrt},$$

$$I_B: \ z^2 \le x; \ y_1 = (z+1)^2; \ y_2 = 2z + 1,$$

$$I_C: \ Q_{sqrt}.$$

Крок 3 – доведення умов локальної коректності. На цьому кроці будуються умови локальної коректності і доводиться їх істинність.

Введемо додаткові позначення. Нехай α – деякий базовий шлях з контрольної точки S у контрольну точку T. Через R_α позначимо формулу, яка істинна при переході через S тоді і тільки тоді, коли обчислення йде з цієї точки далі по шляху α. Ця формула називається умовою вибору шляху α.

Через θ_α позначимо підстановку. Для змінної v вираз $\theta_\alpha(v)$ має значення v після проходження α, виражене через значення змінних перед проходженням α. Підстановка θ_α називається перетворенням змінних, що відповідає шляху α. Вираз $\theta_\alpha(I_K)$ буде означати результат застосування до формули I_K підстановки θ_α , яка відповідає операторам присвоювання шляху α.

Для нашої програми:

R_α : true,

θ_α : $y_1 := 1, \quad y_2 := 1, \quad z := 0,$

R_β : $y_1 \le x$,

θ_β : $y_2 := y_1 + y_2 + 2, \quad y_2 := y_2 + 2, \quad z := z + 1,$

R_γ: $y_1 > x$,

θ_β: – ідентична підстановка.

У теорії Флойда доводиться таке твердження.

Припустимо, що індуктивне твердження для початкової точки є передумова P, і для будь-якого базового шляху α з контрольної точки S у контрольну точку T формула $\{I_S, R_\alpha\} \rightarrow \theta_\alpha(I_T)$ істинна при довільних значеннях змінних. Тоді для будь-якої точки S індуктивне твердження I_S – інваріант в S для передумови P.

Якщо для шляху α вказана формула істинна, то шлях називається локально коректним. Сама формула називається умовою локальної коректності α.

Для нашої програми умови локальної коректності мають вигляд.

Шлях α:

$P_{sqrt} \rightarrow \theta_\alpha(I_B)$,

або $(x \geq 0) \rightarrow \{ 0^2 \leq x,\ 1 = (0+1)^2,\ 1 = 2 \bullet 0 + 1 \}$.

Шлях β:

$\{ I_B,\ y_1 \leq x \} \rightarrow \theta_\beta(I_B)$,

або $\{ z^2 \leq x,\ y_1 = (z+1)^2,\ y_2 = 2z + 1,\ y_1 \leq x \} \rightarrow$

$\rightarrow \{ (z+1)^2 \leq x,\ y_1 + y_2 + 2 = (z+2)^2,\ y_2 + 2 = 2(z+1)+1 \}$.

Шлях γ:

$\{ I_B,\ y_1 > x \} \rightarrow Q_{sqrt}$,

або $\{ z^2 \leq x,\ y_1 = (z+1)^2,\ y_2 = 2z + 1,\ y_1 > x \} \rightarrow$

$\rightarrow \{ z^2 \leq x < (z+1)^2 \}$.

Можна далі показати, що програма завершується для будь-якого невід'ємного **x**. Це гарантує повну коректність програми.

Значний вклад у верифікацію програм було внесено Хоаром. Розглянемо коротко теорію Хоара.

Якщо для кожного оператора за заданими постумовами можна обчислити найслабшу передумову, то для програми можна побудувати коректне доведення. Результат, який треба одержати після виконання програми, береться як постумова останнього оператора. Далі програма відстежується від кінця до

початку з послідовним обчисленням найслабших передумов для кожного оператора.

При досягненні початку програми перша її передумова буде умовою, при якій програма видає необхідний результат.

Для простих операторів обчислення найслабшої передумови може бути задано аксіомою, для більш складних операторів використовуються правила логічного виведення.

Аксіомою називаються логічний вираз, що передбачається правильним.

Правилом логічного виведення називається метод виведення істинності одного твердження на основі значень інших.

Семантику операторів будемо описувати вираженням {P} S {Q}.

Оператор присвоювання

Нехай x := E – оператор присвоювання, а Q – його постумова.

Тоді передумова P визначається аксіомою $P = Q_{x=E}$, це означає, що передумова P обчислюється як умова Q , у якій всі x заміняють виразом E.

Наприклад, для оператора x:= x-3 і постумови {x>0} найслабшою передумовою буде P: {x>3}.

Запис у формі {P} S {Q} має вигляд

{x>3} x:=x-3 {x>0}.

Візьмемо P' = {x>5}, для P' справедлива рівність P' → P.

Справедливий також запис

P' S Q

{x>5} x:=x-3 {x>0} ,

хоча це і не випливає з нашої аксіоми. Для доведення потрібно ще використати правило логічного висновку.

Правила логічного висновку

Загальна форма правила логічного виведення має вигляд

$$\frac{S1, S2, \ldots, Sn}{S}.$$

Це означає, що з істинності S1,…,Sn випливає істинність S.

Правилами логічного виведення є, зокрема, правила логічного висновку:

1) $\dfrac{\{P\}S\{Q\}, P' \to P}{\{P'\}S\{Q\}}$,

2) $\dfrac{\{P\}S\{Q\}, Q' \to Q}{\{P\ \}S\{Q'\}}$.

Іншими словами, правило логічного висновку стверджує, що передумова завжди може бути посилена, а постумова ослаблена.

Послідовність операторів. Нехай S1 і S2 – сусідні оператори програми з наступними передумовами і постумовами:

{P1} S1 {P2}, {P2} S2 {P3}.

Правило виведення для цієї послідовності буде

$$\frac{\{P1\}S1\{P2\}, \{P2\}S2\{S3\}}{\{P1\}S1; S2\{P2\}}.$$

Приклад 3 Нехай S1 і S2 є операторами присвоювання:

y:= 3 * x +1,

x:= y +3,

{x<10}.

Передумова другого оператора {y<7} , а першого – {x<2}.

Умовний оператор. Правило логічного виведення для умовного оператора

$$\frac{\{B\,and\,P\}S1\{Q\}, \{(not\,B)\,and\,P\}S2\{Q\}}{\{P\}\,if\ B\,then\,S1\,else\,S2\{Q\}}.$$

Приклад 4 if x>0 then y:=y -1 else y:=y+1 .

Постумовою оператора є {y>0} .

У разі вибору then використовуємо аксіому присвоювання y:=y -1.

Це дає передумову { y >1 }.

У разі вибору else використовуємо y:=y+1.

Це дає {y>-1}, таким чином, передумова має вигляд {y>1} or {y>-1}, але оскільки {y>1} → {y>-1}, можна записати коротше: {y>-1}.

Цикл з передумовою (while)

У методі індукції існує поняття «індуктивне припущення».

Аналогічну роль у нашому доведені буде відігравати твердження, назване «інваріантом циклу». На його істинність не повинне впливати виконання операторів циклу, тому він і називається інваріантом.

Аксіоматичний опис циклу while:

{P} while B do S {Q}.

Правило логічного виведення для обчислення предумови циклу while виглядає так:

$$\frac{\{I \text{ and } B\}\, S\, \{I\}}{\{I\}\, while\ B\ do\ S\, \{I \text{ and } \{not\ B)\}}\,,$$

де I – інваріант циклу.

Інваріант циклу повинний задовольняти декілька умов.

По-перше, з найслабшої передумови повинна випливати істинність I, по-друге, з істинності I повинна випливати істинність постумови. Оператори циклу не повинні впливати на інваріант, а цикл повинен закінчуватися.

Основні умови у символьному вигляді:

P → I,

{I and B} S {I},

(I and not B) →Q.

Приклад while y < > x then y:=y+1; {y=x}

Для знаходження інваріанта обчислимо ряд передумов для декількох повторень, щоб знайти закономірність.

Для 0 повторень {y=x},

для 1 повторення {y=x-1},

для 2 повторень {y=x-2}.

Таким чином, простежується закономірність {y<=x}, яка і буде інваріантом.

Перевіримо умови:

1) P збігається з I , отже, P→ I ;

2) {I and B} S {I} відповідає {(y<=x) and (y < > x)} y:=y+1 {y<=x}.

Виконавши присвоювання, одержимо {y+1<=x}, що рівносильно умові {y<x}. А ця умова випливає саме з {(y<=x) and (y < > x)} ;

3) (I and (not B))→Q відповідає {(y<=x) and (not(y < > x))}→{y=x}.

У цьому разі передумова є найслабшою, узагалі ж обчислення інваріанта циклу приводить до передумов, що не завжди є найслабшими.

Приклад 6 Обчислення факторіала

```
{n>=0}
k:=n-1;  fact:=n;
while k < > 0 do begin
        fact := fact * k;
        k:=k-1
end;
{fact = n!}.
```

Інваріант циклу – (fact = (k+1) … n) and (k>=0).

Перевіримо умови.

1 Візьмемо I за P, тоді P→I .

2 У виразі {I and B} S {I} підвираз {I and B} означає (fact = (k+1) ... n) and (k>0).

Другий оператор циклу має вигляд {P} k:=k-1 {I},

для нього P: (fact = k … n) and (k>=1).

Перший оператор циклу має вигляд

{P} fact:= fact*k {(fact = k ... n) and (k>=1)},

для нього P: (fact = (k+1) ...n) and (k>=1), іншими словами, P збігається з {I and B}.

3 Перевіримо виконання I and B → Q:

(fac t= (k+1) … n) and (k >= 0) and (k = 0) → fact = n! .[7]

3.3 Алгоритм реалізації тренажеру

На головній сторінці тренажера відображається наступна інформація:

- Назва теми тренажеру.
- Назва дисципліни.
- Інформація про розробника та керівника.

Тренажер містить шість вправ. Користувач може вибрати одну з вправ. Після вибору однієї із вправ на екран виводиться умова завдання та декілька варіантів відповіді. Розглянемо алгоритм прикладів вправ на вибір декількох варіантів відповіді.

Крок 1 Умова завдання:

«Допустимий інтервал значень для числа a та b - від 0 до 10 включно. Для тестування функції порівняння двох чисел на рівність a == b необхідні такі приклади»:

- **a = 3, b = 5, значення функції = false (1)**
- **a = 3, b = 3, значення функції = true (2)**
- a = 0, b = 0, значення функції = true **(3)**
- a = 10, b = 10, значення функції = true **(4)**
- **a = 0, b = 10, значення функції = false (5)**
- **a = 10, b = 0, значення функції = false (6)»**

Треба вибрати декілька вірних відповідей.

Якщо вибрана 1,2,5 та 6 приклади, відповідь зараховується і користувач може перейти на наступній крок, якщо ж відповідь не вірна, на екрані відображається підказка, причому користувач залишається на цьому ж вікні для повторного вибору відповіді.

Крок 2 Умова завдання:

«Допустимий інтервал значень для числа a та b - від 0 до 10 включно. Для тестування функції порівняння двох чисел на рівність a < b необхідні такі приклади:

- **a = 5, b = 6, значення функції = true (1)**
- **a = 5, b = 4, значення функції = false (2)**
- **a = 6, b = 6, значення функції = false (3)**
- a = 0, b = 0, значення функції = false **(4)**
- **a = 0, b = 10, значення функції = true (5)**
- **a = 10, b = 0, значення функції = false (6)**
- a = 10, b = 10, значення функції = false **(7)**»

Треба вибрати декілька вірних відповідей.

Якщо вибрана 1,2,3,5 та 6 приклади, відповідь зараховується і користувач може перейти на наступній крок, якщо ж відповідь не вірна, на екрані відображається підказка, причому користувач залишається на цьому ж вікні для повторного вибору відповіді.

Крок 3 Умова завдання:

«Допустимий інтервал значень для числа a та b - від 0 до 10 включно. Для тестування функції порівняння двох чисел на рівність a >= b необхідні такі приклади:

- **a = 5, b = 6, значення функції = false (1)**
- **a = 5, b = 4, значення функції = true (2)**
- **a = 6, b = 6, значення функції = true (3)**
- a = 0, b = 0, значення функції = true **(4)**
- **a = 0, b = 10, значення функції = false (5)**
- **a = 10, b = 0, значення функції = true (6)**
- a = 10, b = 10, значення функції = true **(7)**»

Треба вибрати декілька вірних відповідей.

Якщо вибрана 1,2,3,5 та 6 приклади, відповідь зараховується і користувач може перейти на наступній крок, якщо ж відповідь не вірна, на екрані відображається підказка, причому користувач залишається на цьому ж вікні для повторного вибору відповіді.

Розглянемо алгоритм прикладів вправ на вибір однієї вірної відповіді.

Крок 1 Умова завдання:

Які мінімальні набори тестових прикладів можна використовувати для повного покриття наступної ділянки програмного коду по гілках?

```
if (a == 0) {          call_1();}
else {       if (b > 0) call_2();}
```

Варіанти відповідей:

- **a = 0, b = 0; 2) a = 1, b = 1; 3) a = 1; b = 0 (1)**
- a = 0, b = 0; 2) a = 1, b = 1**(2)**
- a = 0, b = 1; 2) a = 0, b = 0; 3) a = 1; b = 1**(3)**
- a = 0, b = 1; 2) a = 0, b = 1; 3) a = 0; b = 0**(5)**

Треба вибрати один з варіантів.

Якщо вибрано приклад під номером 1 відповідь зараховується і користувач може перейти на наступній крок, якщо ж відповідь не вірна, на екрані відображається підказка, причому користувач залишається на цьому ж вікні для повторного вибору відповіді.

Крок 2 Умова завдання:

Які мінімальні набори тестових прикладів можна використовувати для повного покриття наступної ділянки програмного коду за рядками та умовами?

```
if ( (a == 0) && (b = 0) && (c == 1) {         call_1();}
 else
{ if (d == 1) call_2()}
```

Варіанти відповідей:

- a = 0, b = 0, c = 1, d = 1; 2) a = 1, b = 1, c = 0, d = 0 **(1);**
- **a = 0,b=0,c = 1, d = 1;2) a=1, b=1, c=0, d=0;3) a=1, b=1, c=0, d=1(2)**
- a=1, b=0, c=1, d=1; 2)a=1, b=1, c=0, d=0; 3)a=1, b=1, c=0, d=1**(3)**
- a=0, b=0, c=1, d=0; 2)a=1, b=1, c=0, d=0; 2)a=1, b=1, c=0, d=0**(4)**

Треба вибрати один з варіантів.

Якщо вибрано приклад під номером 2 відповідь зараховується і користувач може перейти на наступній крок, якщо ж відповідь не вірна, на екрані відображається підказка, причому користувач залишається на цьому ж вікні для повторного вибору відповіді.

Крок 3 Умова завдання:

Які мінімальні набори тестових прикладів можна використовувати для повного покриття наступної ділянки програмного коду за рядками та умовами?

```
if ( (a == 0) && (b = 0) && (c == 1)
{ if (d == 1) call_1();}
 else
{ call_2()}
```

Варіанти відповідей:

- 1)a = 0, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0 **(1)**
- **1) a=0, b=0, c=1, d=1; 2) a=0, b=0, c=1, d=0; 2) a=1, b=1, c=0, d=1(2)**
- 1) a=1, b=0, c=1, d=1; 2) a=1, b=1, c=0, d=0; 2) a = 1, b = 1, c = 0, d = 1**(3)**
- 1) a=0, b=0, c=1, d = 0; 2)a=1, b=1, c=0, d=0; 2)a=1, b=1, c=0, d=0**(4)**

Треба вибрати один з варіантів.

Якщо вибрано приклад під номером 2 відповідь зараховується і користувач може перейти на наступній крок, якщо ж відповідь не вірна, на екрані відображається підказка, причому користувач залишається на цьому ж вікні для повторного вибору відповіді.

3.4 Розробка блок-схеми

На рис 3.2 зображено блок-схему алгоритму роботи тренажеру.

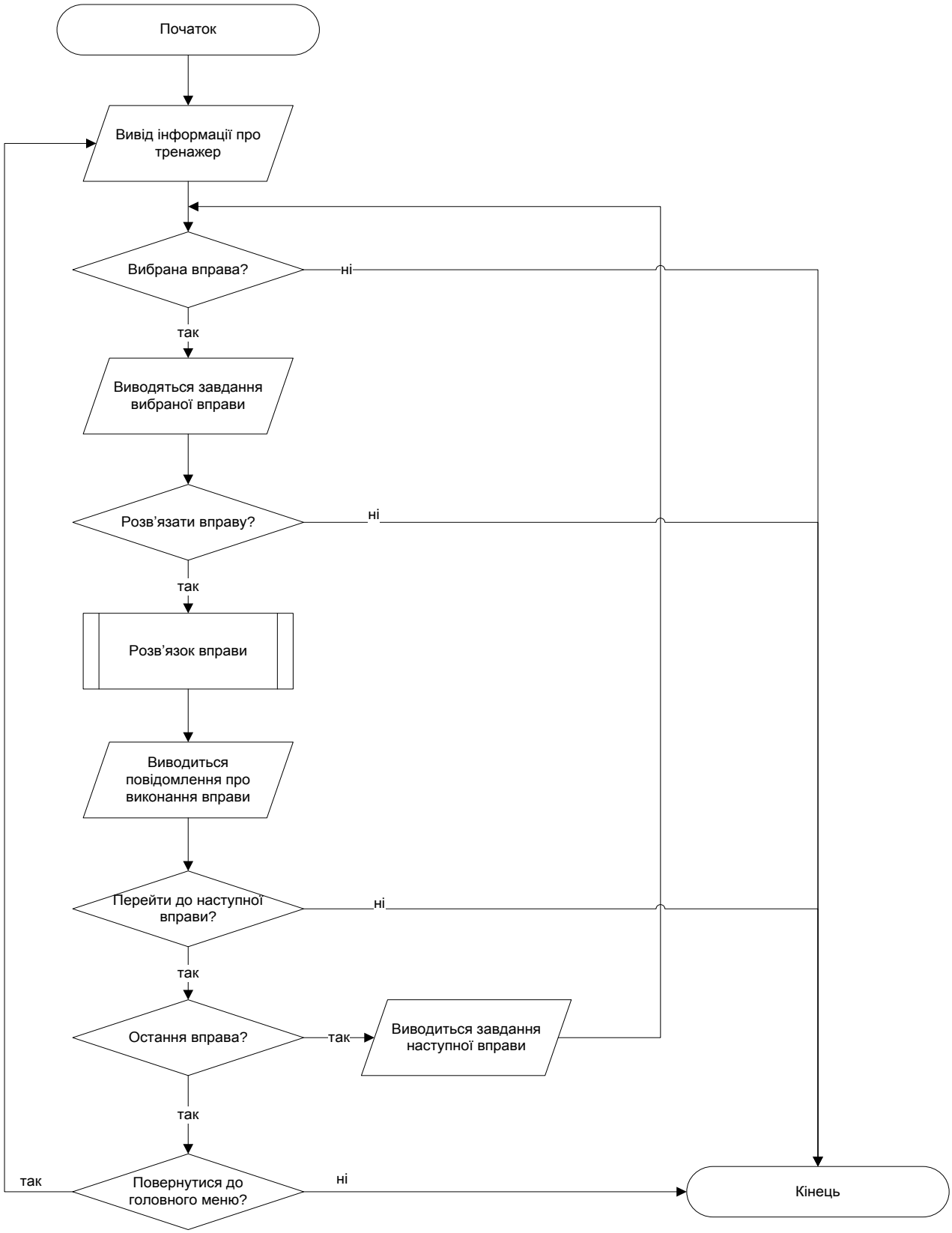

Рисунок 3.2- Блок-схема алгоритму роботи тренажеру

На рис. 3.3 зображена блок-схема розв'язання вправ на вибір декількох відповідей.

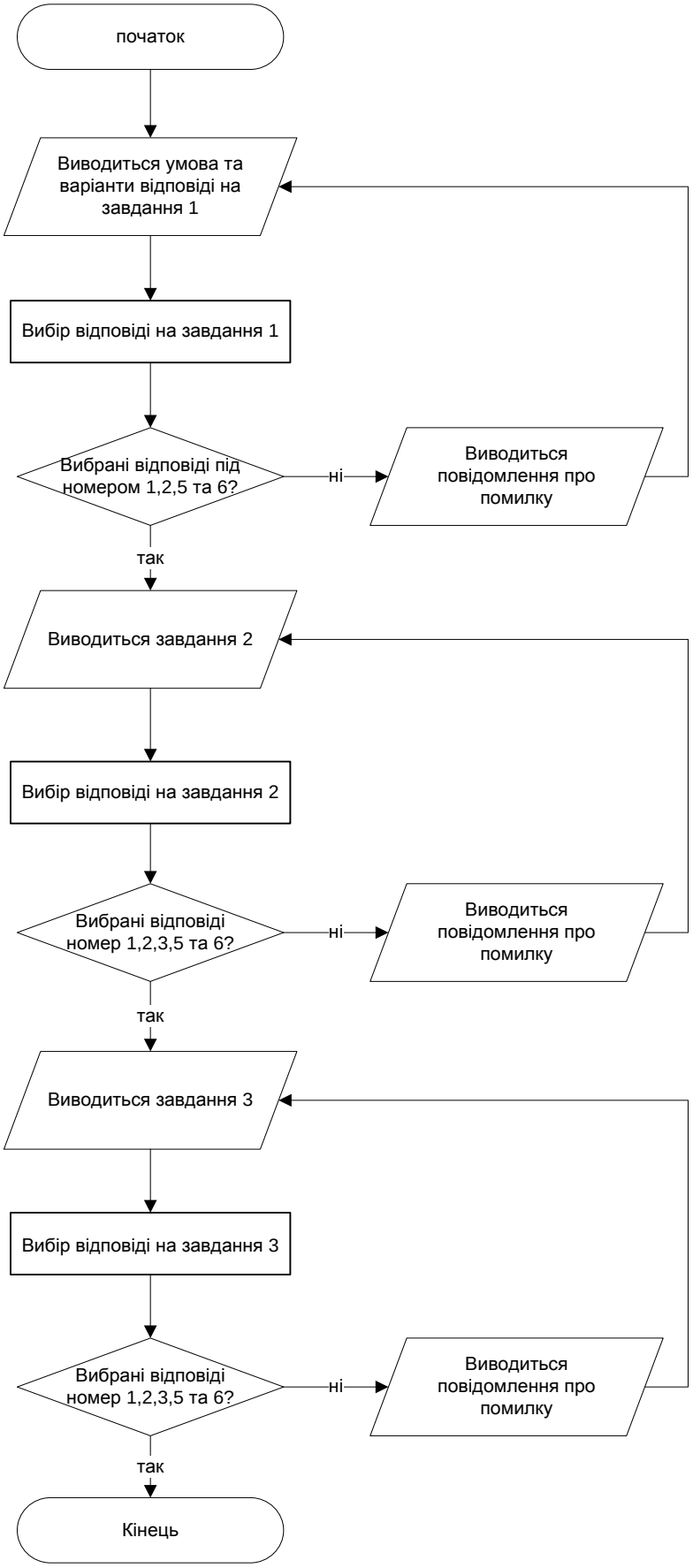

Рисунок 3.3 - блок-схема розв’язання вправ на вибір декількох варіантів відповідей

На рис. 3.4 зображена блок-схема розв’язання вправ на вибір однієї відповіді.

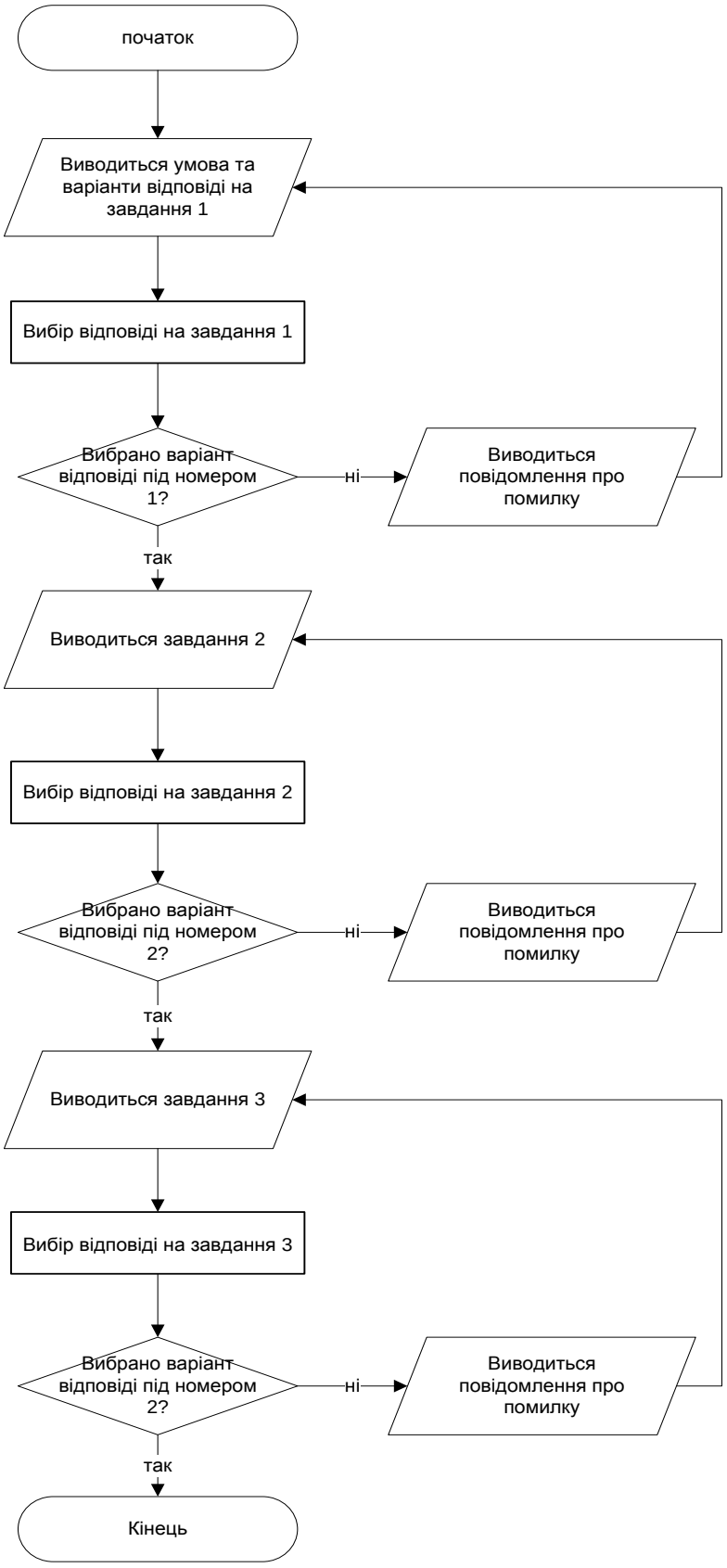

Рис. 3.4 - блок-схема розв’язання вправна вибір однієї відповіді

4. ПРАКТИЧНА ЧАСТИНА

4.1 Платформа та засоби розробки системи.

Оскільки тренажер «Верифікація програм» є веб-додатком, то для реалізації дизайну сайту використані технології та встановлено таке програмне забезпечення:

1. **Bootstrap** – це відкритий та безкоштовний HTML, CSS та JS фреймворк, який використовується веб-розробниками для швидкого створення адаптивних дизайнів сайтів.

Bootstrap – це набір файлів (CSS та JavaScript). Після підключення цих файлів до сторінки становиться доступними для верстки дизайну велика кількість класів та готових компонентів. Використовуючи їх можна дуже швидко та якісно створити сучасний адаптивний дизайн сайту. [17]

2. **Reakt** – це JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів користувача. Він має вбудовану мову, яка базується на HTML.

Reakt складається з:

- ✓ <Component> - React дозволяє визначити компоненти як класи або функції.
- ✓ <JSX> – розширення мови JavaScript. Це абстракція, яка дозволяє використовувати синтаксис HTML усередині коду JavaScript і за допомогою якої ви можете створювати компоненти React, які виглядають як стандартна HTML-розмітка.
- ✓ < localStorage > - це веб-сховища браузера. Він дозволяє користувачам зберігати дані у вигляді пар ключ/значення у браузері для подальшого використання. [18]

Для реалізації функціоналу додатку було вибрано мову програмування **JavaScrip**t. Ця мова динамічна та об’єктна-орієнтована. Її найчастіше використовують для створення сценаріїв веб-сторінок, що надає можливість на

боці клієнта взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінок [10].

Для участі в розробці встановлено програмну платформу **Node.js** – це середовище виконання JavaScript. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виводу через свій API, підключати інші зовнішні бібліотеки, написані різними мовами, забезпечуючи виклики до них з JavaScript-коду. В основі Node.js лежить обєктно-орієнтоване та асинхронне (або реактивне) програмування з неблокуючим введенням/виводом [19]. Посилання на **Node.js** версії

Для написання та редагування програмного коду та для побудови і налагодження тренажеру було обрано редактор коду **Visual Studio Code**, який дозволяє розробляти як консольні програми, так і веб-сайти, веб-програми, веб-служби для всіх платформ. У редакторі присутні інструменти для роботи з Git. Він підтримує розробку для платформ ASP.NET і Node.js, і вважається легковажним рішенням, яке дозволяє обійтися без повного інтегрованого середовища розробки. Великим плюсом редактора є підтримка великої кількості мов у тому числі і JavaScript.

Вільний доступ до інтерфейсу було реалізовано за допомогою сучасного сервісу безкоштовного сервісу **GitHub** – це система управління проектами і версіями програмного коду. Посилання на проект в **GitHub .**

4.2 Процес програмної реалізації

Розробку розпочато зі створення папки та структури проекту (Рис. 4.1)

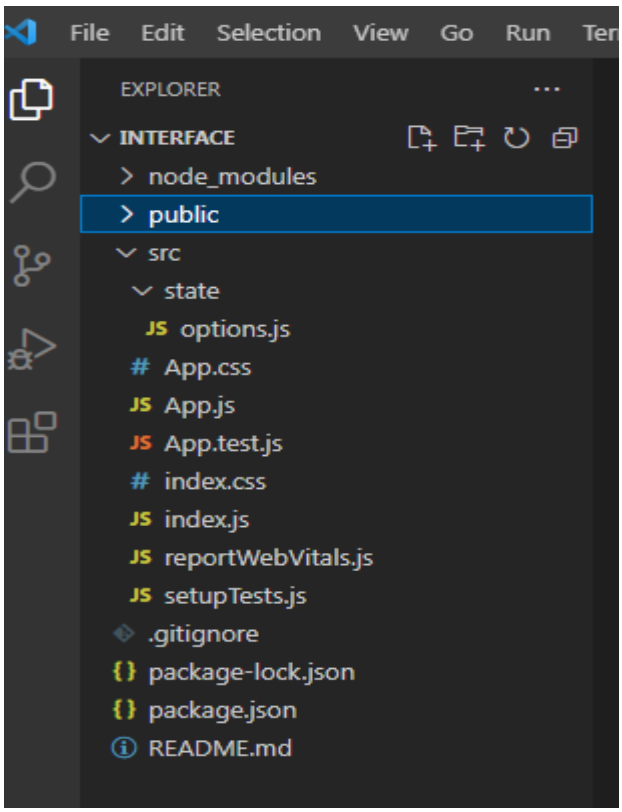

Рисунок 4.1 - Структура проекту

На першому етапі створення проекту в папку **Node_mobules** за допомогою Node Package Manager (NPM) були встановлені пакети необхідні для розробки інтерфейсу.

SRC – це робоча папка проекту, яка містить наступні файли:

- **App.css** – це файл де описані стилі інтерфейсу (Додаток А).
- **Options.js** - містить опис завдань та варіанти відповідей (Додаток Б). Так як тренажер призначено для англомовного курсу, всі завдання та варіанти відповідей перекладені на англійську мову.
- **App.js** – основний компонент інтерфейсу(Додаток В), який містить наступні елементи програмного коду:
    - Імпорт сталей, із файлу **App.css,** на рис.4.2 зображена команда імпорту

```
1 import './App.css';
```

Рисунок 4.2 - Команда імпорту стилів з файлу **App.css**

- Імпорт таких елементів як слайди та кнопки із **react-bootstrap** на рис.4.3 зображена команда імпорту.

```
2   import { Carousel, Button } from 'react-bootstrap';
```

Рисунок 4.3 - Команда імпорту слайдів та кнопки з react-bootstrap

- Імпорт завдань з **./state/options, на р**ис.4.4 зображена командаімпорту

```
3   import { allTasks } from './state/options';
```

Рисунок 4.4 - Команда імпорту завдань з **./state/options**

- функція **App()** – відкриває слайди із завданням та варіантами відповідей та створює завдання яке імпортує з папки **./state/options** та для кожного завдання створює слайд. На рис. 4.5 функція **App()**;

```
function App() {
  return (
    <div className="App">
      <div className='title-page'>
        <h2>Training apparatus</h2>
        <div className = "sub-title">from the topic</div>
        <h2>Verification of programs</h2>
        <div className = "sub-title">disciplines</div>
        <h2>Theory of programming</h2>
        <a href="#slider">Start training</a>
        <div className = "author-info">
          <span>Author: </span><p>Сімперович М.М.</p> <br />
          <span>Lead: </span><p>Черненко О.О.</p>
        </div>
        <p style={{'margin-top': '20px'}} >ВНЗ Укоопспілки "Полтавський університет економіки і торгівлі"</p>
        <p>ПОЛТАВА-2022</p>
      </div>
      <Carousel id="slider" variant="dark" interval={null}>
        {allTasks.map((task) => {
          return (
            <Carousel.Item id = {task.title}>
            <Carousel.Caption>
              <h3 style={{'margin-bottom': '120px'}} id = {task.id}>{task.title}</h3>
              <fieldset style={{'margin-bottom': '120px'}}>
                {task.options.map((option) => {
                  return (<div key={option.optionValue} className = "anwer-wrapper">
                    <label htmlFor={option.optionValue}>{option.optionValue}</label>
                    <input type="checkbox" name={option.optionValue} id={option.optionValue}/>
                  </div> )
                  })}
              </fieldset>
              <Button style={{'margin-bottom': '20px'}} onClick={handleAnswer}>Відповісти</Button>
                <span>Wrong checked checkbox will be colored in red</span>
              </Carousel.Caption>
            </Carousel.Item>
            )
          })}
        </Carousel>
      </div>
    );
  }

  export default App;
```

Рисунок 4.5 - Функція **App()**

- функція **HendLeAnswer** обробляє правильність відповіді. На рис. 4.6 зображена функція **HendLeAnswer**

```
const handleAnswer = () => {
  const selectedCheckboxes = Array.from(document.querySelectorAll('.active.carousel-item input[type="checkbox"]:checked')
  const selectedCheckboxesIds = selectedCheckboxes.map((checkbox) => checkbox.id);
  const activeTaskId = document.querySelector('.active.carousel-item h3').id;
  const currentTask = allTasks.find((task) => task.id === activeTaskId);
  const numberOfTrues = currentTask.options.filter((option) => option.isTrue).length;

  const wrongCheckboxes = currentTask.options.filter((option) => {
    if(selectedCheckboxesIds.includes(option.optionValue) && !option.isTrue) {
      return option;
    }
  })

  document.querySelectorAll('.active.carousel-item label').forEach((task) => {
    if(wrongCheckboxes.map((wrongTask) => wrongTask.optionValue).includes(task.htmlFor)) {
      console.log(task);
      task.style.color = 'red';
    };
  });

  if(wrongCheckboxes.length || numberOfTrues != selectedCheckboxesIds.length) {
    alert('Answer is false');
    return;
  }

  return alert('Answer is true');
}
```

Рисунок 4.6 - Функція **HendLeAnswer ()**

- **Package.json** – містить опис всіх бібліотек, які були використані при розробці інтерфейсу (Додаток Г).

4.3 Опис роботи програмного забезпечення

Після запуску програми на екрані з’являється титульний лист на якому на англійській мові виведена наступна інформація:

- назва теми тренажеру;
- назва дисципліни;
- прізвище розробника та керівника проекту

На рис. 4.7 зображена початкова сторінка тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування».

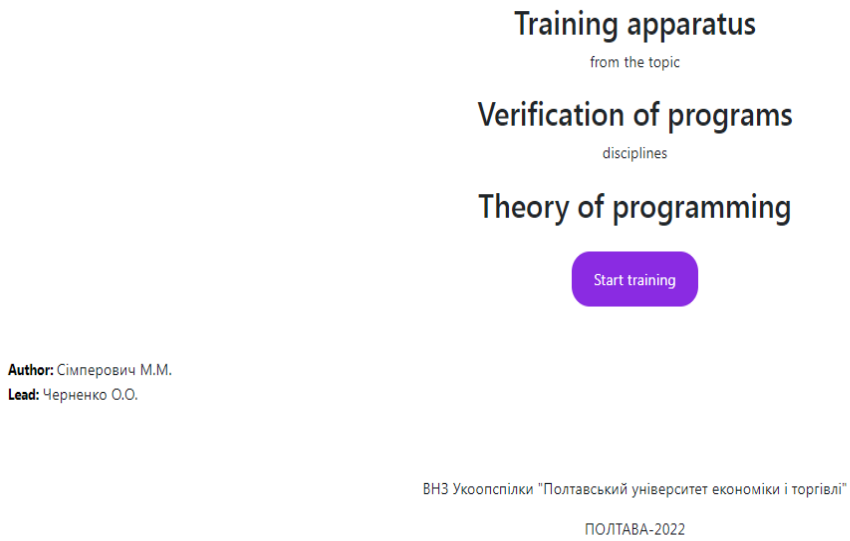

Рисунок 4.7 - Початкова сторінка тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування»

Для початку виконання завдання треба натиснути на кнопку «Start training». На екрані з’явиться слайд з першим завдання та варіантами відповідей до нього (рис. 4.8).

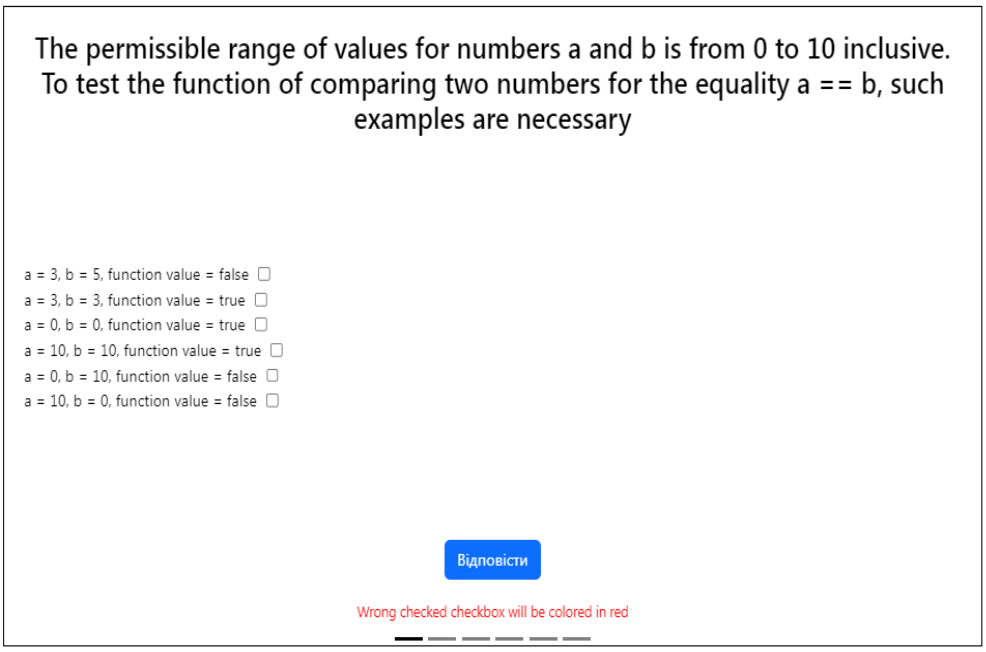

Рисунок 4.8 - Слайд першого завданням тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування»

Студент обирає декілька можливих відповідей на запитання, якщо якась з відповідей не вірна на екрані з’являється повідомлення, про те що у відповіді є помилка не вірні відповіді при цьому зафарбовуються у червоний колір. На рис. 4.9 продемонстроване вікно у випадку невірної відповіді.

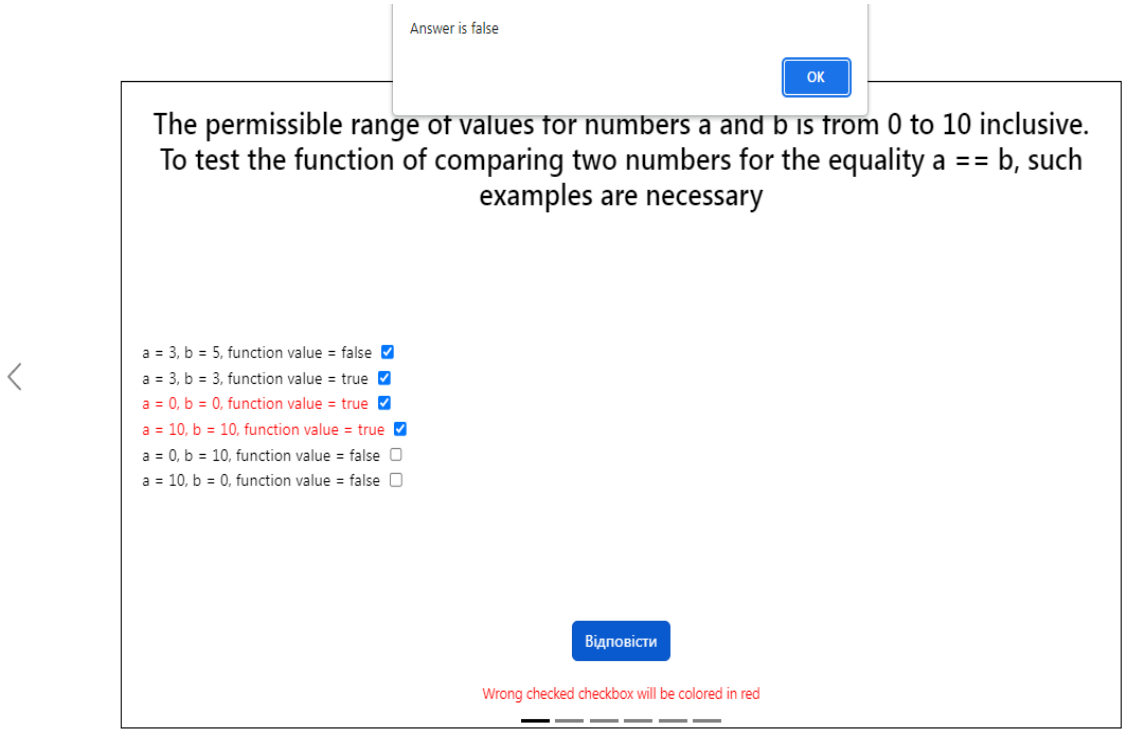

Рисунок 4.9 - Слайд першого завданням тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування» у випадку не вірної відповіді

Якщо відповідь вірна на екрані з’являється повідомлення, зображене на рис.4.10

Рисунок 4.10 - Повідомлення при вірній відповіді.

Після натискання на кнопку «ОК» переходимо на наступний слайд на якому міститься питання та варіанти відповідей до другого завдання. Принцип роботи кожного наступного слайду аналогічний. На рис. 4.11 – 4.15 зображено слайди з наступними завданнями тренажеру.

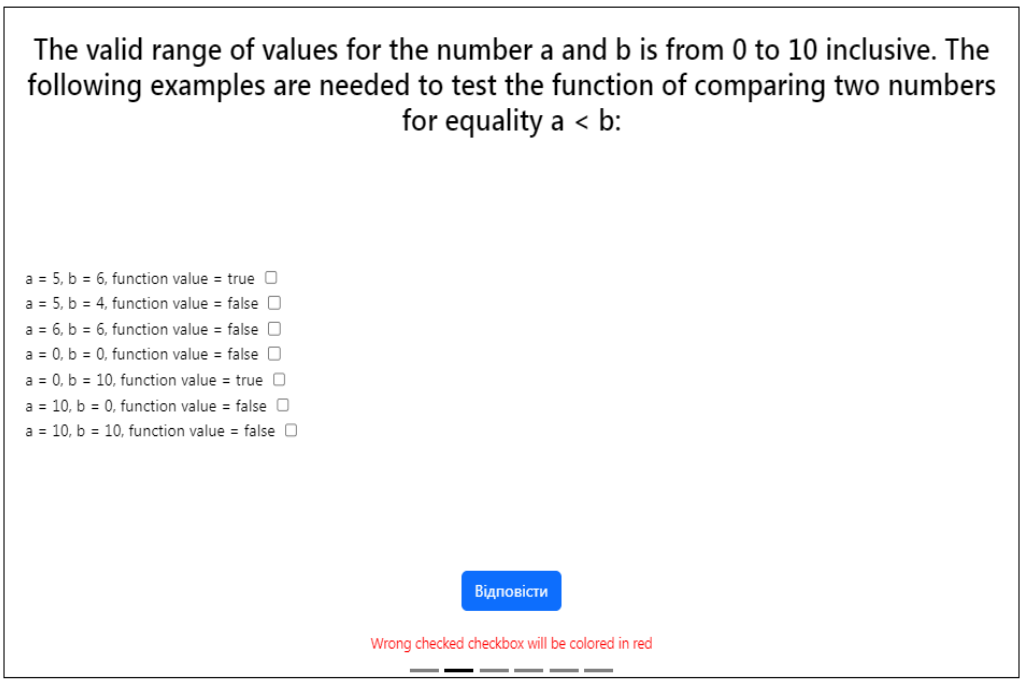

Рисунок 4.11 - Слайд другого завдання тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування»

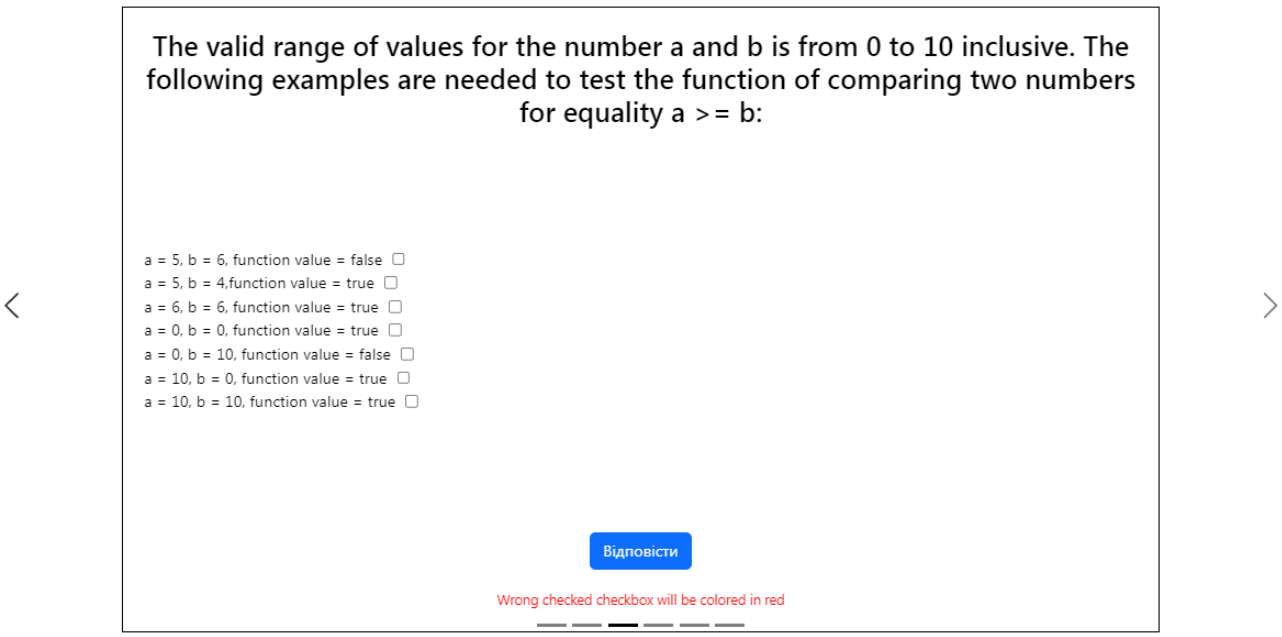

Рисунок 4.12- Слайд третього завдання тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування»

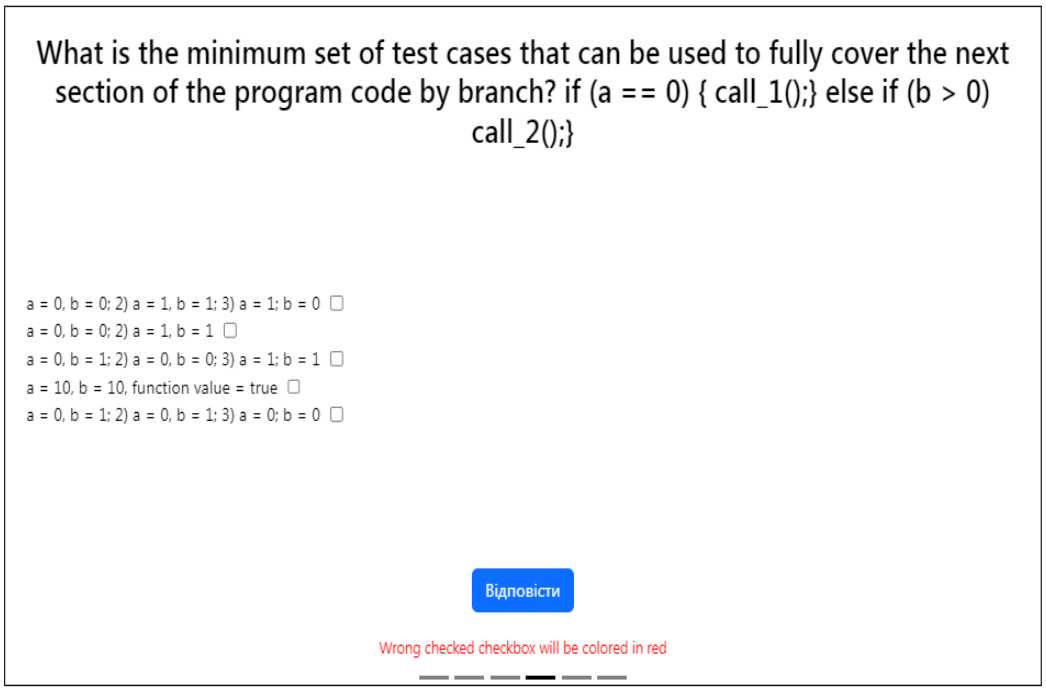

Рисунок 4.13 - Слайд четвертого завдання тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування»

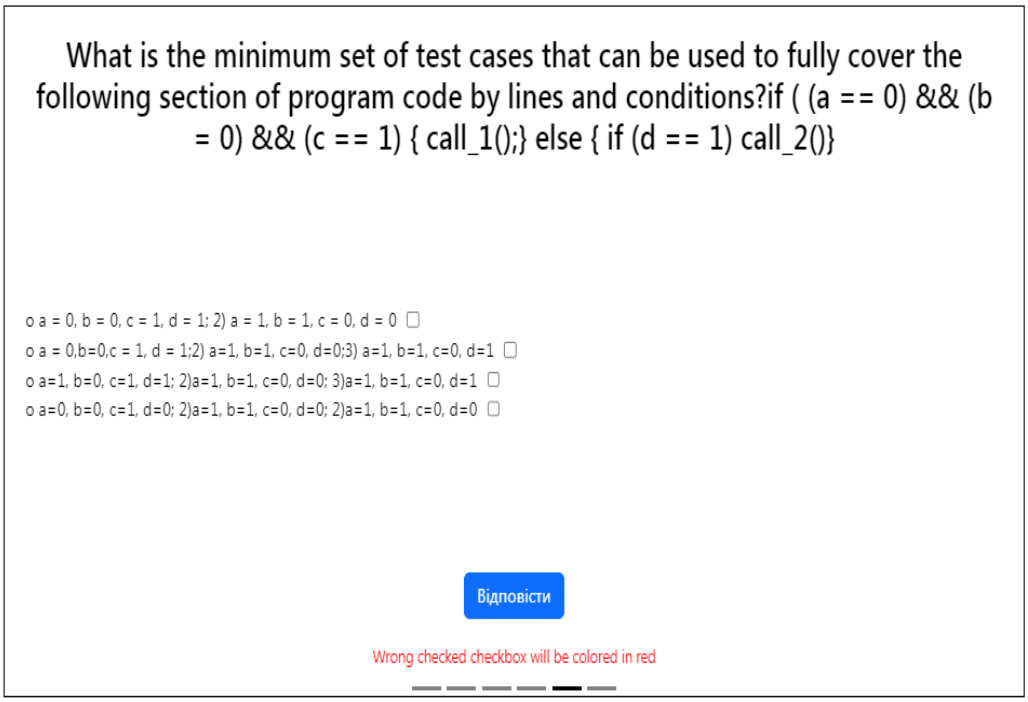

Рисунок 4.14 - Слайд п’ятого завдання тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування»

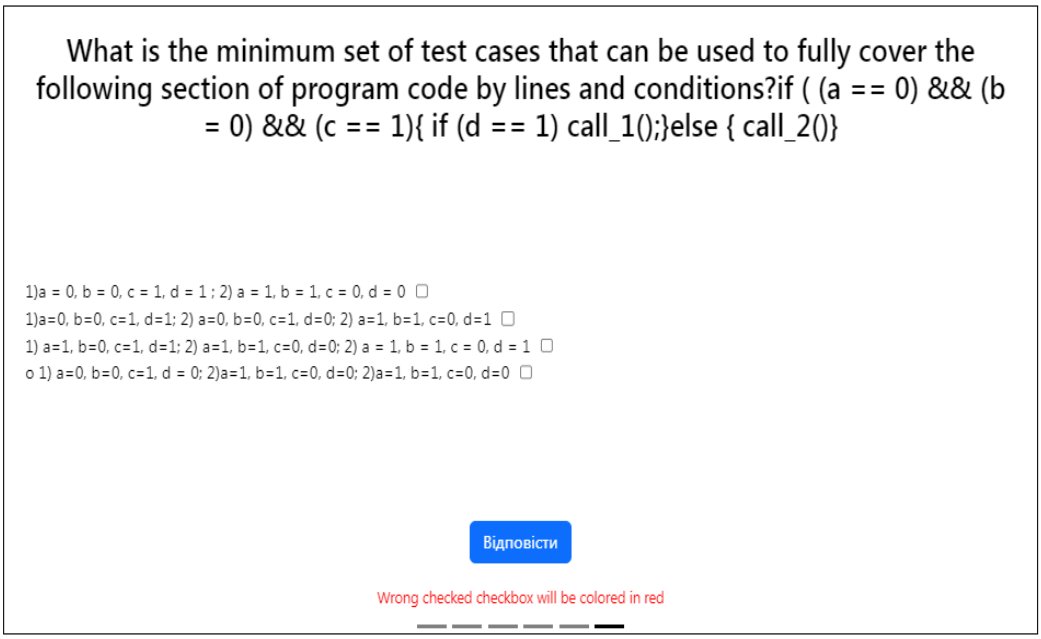

Рисунок 4.15 - Слайд шостого завдання тренажеру з теми «Верифікація програм» з дисципліни «Теорія програмування»

Після виконання всіх завдань тренажеру, для закріплення знань, їх можна пройти знову.

ВИСНОВКИ

Мета дипломної роботи полягала у розробці тренажеру з теми «Верифікація програм» англомовного дистанційного навчального курсу «Теорія програмування».

При виконанні роботи було:

- проаналізовано навчальні тренажери з курсу «Теорія алгоритмів» - тренажери на тему «1R - алгоритм», «Аналіз алгоритму сортування бульбашками» та «Аналіз алгоритму швидкого сортування»;
- досліджено тему «Верифікація програм»;
- підібрані приклади на цю тему;
- розроблено алгоритм за яким створені блок схеми реалізації алгоритму;
- розроблено ПЗ навчального тренажер;
- виконане тестування та корегування ПЗ.

Для реалізації проекту було вирішено використати технології для розробки веб – доданків, тому що дана технологія дозволяє забезпечити загальнодоступність створюваних засобів. Технологічно це призводить до необхідності реалізації тренажерів у вигляді веб-прикладів, які виконуються або на стороні сервера, або на стороні клієнта. Тому для реалізації дизайну сайту вибрані такі технології як: **Bootstrap, Reakt.** Для реалізації функціоналу додатку було вибрано мову програмування **JavaScrip**t. Для написання та редагування програмного коду та для побудови і налагодження тренажеру було обрано редактор коду **Visual Studio Code.**

Результатом виконання дипломного проекту є розробка веб-інтерфейса навчального тренажеру, який адаптований для англомовних студентів. На головній сторінки користувачу відображається інформація, щодо назви теми тренажеру та курсу , його розробника та керівника проекту. Тренажер містить два види завдань, по три кожного виду. Після вибору завдання на кожному кроці відображається умова завдання та варіанти відповідей.

Програма надає користувачеві наступні основні функції:

- тренажер розроблений англійською мовою, що значно спрощує роботу з ним іноземним студентам;
- вибір завдань;
- студенти можуть наглядно розібрати задачі з кожної вправи стільки раз, скільки їм потрібно для розуміння матеріалу.
- якщо відповідь була не вірною, система дає підказку.
- Програмний код проекту написано з застосування техніки ООП, що дозволяє легко змінювати програмний код для подальшої модифікації продукту та внесення змін.

Результати роботи опубліковані в збірнику магістрів ПУЕТ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мухамадиева К. Б. Применение тренажеров в системе дистанционного обучения / К. Б. Мухамадиева / Молодой ученый. — 2014. — № 17 (76). — Режим доступу: https://moluch.ru/archive/76/12802/ .

2. Краснова Т. И. Повышение качества обучения в вузе в условиях интеграции смешанной модели обучения / Т. И. Краснова // Молодой ученый. - 2015 - №5(85).—Режим доступу: https://moluch.ru/archive/85/15950/.

3. Лекции лауреатов премии Тьюринга. - М.: Мир, 1993.

4. Миронов А. М. Верификация программ методом инвариантов, Интеллектуальные системы. / А. М. Миронов //Теория и приложения, 2017, том 21, выпуск 4, 31–49с.

5. Лавріщева К.М. Програмна інженерія./ К.М. Лавріщева // К.– 2008.– 319 с.

6. Андерсон Р. Доказательство правильности программ/ Р. Андерсон: пер. с англ. Б. Н. Зобниной // М.: Мир, 1982 168 с.

7. Учебные материалы для студентов. Верификация программ. Семантики языков программирования // Wikipedia. – 2018. – Режим доступу: https:// https://vunivere.ru/work23796.

8. Якименко О.В. Применение обучающих программ-тренажеров в обучении программированию / О.В. Якименко, Стась А.Н. // Вестник ТГПУ. 2019. Выпуск 1.

9. Журнал «Фундаментальные исследования» / 2013 – №10 (часть 12) . – 2778-2781сс.

10. Дебольт. HTML и CSS. Совместное использование / Дебольт // Вирджиния. - М.: НТ Пресс, 2017. - 512 с.

11. Прохоренок Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. – Москва: СПб. [и др.]: Питер, 2015. - 768 с.

12. Фримен Эрик. Изучаем программирование на JavaScript / Фримен

Эрик, Робсон Элизабет - ,2015. – 656 с.

13. Дєвид Флєнаган. JavaScript полное руководство/ Дєвид Флєнаган. – Диалектика – Киев, 2021. – 722с.

14. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений 3-е приложение. Вильямс, 2020. – 720 с.

15. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для академического бакалавриата / С. В. Зыков. – М. : Издательство Юрайт, 2019. — 155 с..

16. Сысолетин, Е. Г. Разработка интернет-приложений : учеб. пособие для вузов / Е. Г. Сысолетин, С. Д. Ростунцев ; под науч. ред. Л. Г. Доросинского. — М. : Издательство Юрайт, 2019. — 90 с.

17. Електронний підручний по HTML, CSS, JS . Режим доступу: https://www.w3schools.com/css/default.asp

18. Самоучитель HTML, CSS. Режим доступу: http://htmlbook.ru/

19. Електронний підручник по JavaScript. Режим доступу: https://learn.javascript.ru/.

20. WebForMySelf. Все о создании сайтов. Режим доступу: https://learn.javascript.ru/

21. Современный интерфейсный фреймворк на основе Material Design. Режим доступу: https://materializecss.com/

22. Веб-технологии для разработчиков. Режим доступу: https://developer.mozilla.org/ru/docs/Web.

23. Способи тестування програмного забезпечення. Режим доступу: https://habr.com/ru/company/otus/blog/443418/

ДОДАТОК А

APP.CSS

```
.title-page{
  height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: flex-start;
  padding-top: 40px;
}
.carousel {
  height: 700px;
}
.carousel-item {
  height: 700px;
}

span {
  display: block;
  color: red;
  font-size: 14px;
}
.author-info {
  width: 100%;
  padding: 50px;
}

.author-info span {
  display: inline;
  color: black;
  font-weight: 800;
}

.author-info p {
  display: inline;
}

.title-page a {
  color: #fff;
  border-radius: 20px;
  background-color: blueviolet;
  text-decoration: none;
  align-items: center;
  padding: 15px 25px;
  margin-top: 20px;
}

.anwer-wrapper {
  display: flex;
  align-items: center;
}

.anwer-wrapper label {
```

```
  padding-right: 10px;
}

.carousel-caption {
  border: 2px solid;
  padding: 20px;
}

.sub-title {
  margin-bottom: 20px;
}
```

ДОДАТОК Б

OPTIONS.JS

```
const store = {
    task1: [
                {
                    id: '1',
                    title: 'The permissible range of values for numbers a and b is from 0 to
10 inclusive. To test the function of comparing two numbers for the equality a == b, such
examples are necessary',
                    options: [
                        {
                            optionValue: 'a = 3, b = 5, function value = false',
                            isTrue: true
                        },
                        {
                            optionValue: 'a = 3, b = 3, function value = true',
                            isTrue: true
                        },
                        {
                            optionValue: 'a = 0, b = 0, function value = true'
                        },
                        {
                            optionValue: 'a = 10, b = 10, function value = true'
                        }
                        ,
                        {
                            optionValue: 'a = 0, b = 10, function value = false',
                            isTrue: true
                        },
                        {
                            optionValue: 'a = 10, b = 0, function value = false',
                            isTrue: true
                        }
                    ]
                },
                {
                    id: '2',
                    title: 'The valid range of values for the number a and b is from 0 to 10
inclusive. The following examples are needed to test the function of comparing two numbers
for equality a < b:',
                    options: [
                        {
                            optionValue: 'a = 5, b = 6, function value = true',
                            isTrue: true
                        },
                        {
                            optionValue: 'a = 5, b = 4, function value = false',
                            isTrue: true
                        },
                        {
                            optionValue: 'a = 6, b = 6, function value = false',
                            isTrue: true
                        },
```

```
                {
                    optionValue: 'a = 0, b = 0, function value = false'
                }
                ,
                {
                    optionValue: 'a = 0, b = 10, function value = true',
                    isTrue: true
                }
                ,
                {
                    optionValue: 'a = 10, b = 0, function value = false',
                    isTrue: true
                }
                ,
                {
                    optionValue: 'a = 10, b = 10, function value = false'
                }
            ]
        },
        {
            id: '3',
            title: 'The valid range of values for the number a and b is from 0 to 10
inclusive. The following examples are needed to test the function of comparing two numbers
for equality a >= b:',
            options: [
                {
                    optionValue: 'a = 5, b = 6, function value = false',
                    isTrue: true
                },
                {
                    optionValue: 'a = 5, b = 4,function value = true',
                    isTrue: true
                },
                {
                    optionValue: 'a = 6, b = 6, function value = true',
                    isTrue: true
                },
                {
                    optionValue: 'a = 0, b = 0, function value = true'
                }
                ,
                {
                    optionValue: 'a = 0, b = 10, function value = false',
                    isTrue: true
                }
                ,
                {
                    optionValue: 'a = 10, b = 0, function value = true',
                    isTrue: true
                }
                ,
                {
                    optionValue: 'a = 10, b = 10, function value = true'
                }
            ]
        }
```

```
    ],
    task2: [
                {
                    id: '4',
                    title: 'What is the minimum set of test cases that can be used to fully
cover the next section of the program code by branch? '+
                    'if (a == 0) {  call_1();} '+
                    'else '+
                    'if (b > 0) call_2();}',
                    options: [
                        {
                            optionValue: 'a = 0, b = 0; 2) a = 1, b = 1; 3) a = 1; b = 0',
                            isTrue: true
                        },
                        {
                            optionValue: 'a = 0, b = 0; 2) a = 1, b = 1'
                        },
                        {
                            optionValue: 'a = 0, b = 1; 2) a = 0, b = 0; 3) a = 1; b = 1'
                        },
                        {
                            optionValue: 'a = 10, b = 10, function value = true'
                        }
                        ,
                        {
                            optionValue: 'a = 0, b = 1; 2) a = 0, b = 1; 3) a = 0; b = 0'
                        }

                    ]
                },
                {
                    id: '5',
                    title: 'What is the minimum set of test cases that can be used to fully
cover the following section of program code by lines and conditions?'+
                            'if ( (a == 0) && (b = 0) && (c == 1) { call_1();} '+
                            'else '+
                            '{  if (d == 1) call_2()}',
                    options: [
                        {
                            optionValue: 'o a = 0, b = 0, c = 1, d = 1; 2) a = 1, b = 1, c =
0, d = 0'
                        },
                        {
                            optionValue: 'o a = 0,b=0,c = 1, d = 1;2) a=1, b=1, c=0, d=0;3)
a=1, b=1, c=0, d=1',
                            isTrue: true
                        },
                        {
                            optionValue: 'o a=1, b=0, c=1, d=1; 2)a=1, b=1, c=0, d=0; 3)a=1,
b=1, c=0, d=1'
                        },
                        {
                            optionValue: 'o a=0, b=0, c=1, d=0; 2)a=1, b=1, c=0, d=0; 2)a=1,
b=1, c=0, d=0'
                        }
```

```
                ]
            },
            {
                id: '6',
                title: 'What is the minimum set of test cases that can be used to fully
cover the following section of program code by lines and conditions?'+
                'if ( (a == 0) && (b = 0) && (c == 1)' +
                '{  if (d == 1) call_1();}' +
                'else '+
                '{  call_2()}',
                options: [
                    {
                        optionValue: '1)a = 0, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c
= 0, d = 0'
                    },
                    {
                        optionValue: '1)a=0, b=0, c=1, d=1; 2) a=0, b=0, c=1, d=0; 2)
a=1, b=1, c=0, d=1',
                        isTrue: true
                    },
                    {
                        optionValue: '1) a=1, b=0, c=1, d=1; 2) a=1, b=1, c=0, d=0; 2) a
= 1, b = 1, c = 0, d = 1',
                    },
                    {
                        optionValue: 'o 1) a=0, b=0, c=1, d = 0; 2)a=1, b=1, c=0, d=0;
2)a=1, b=1, c=0, d=0'
                    }

                ]
            }
    ]
}

export const allTasks = store.task1.concat(store.task2);

export const score = 0;
```

ДОДАТОК В

APP.JS

```
import './App.css';
import { Carousel, Button } from 'react-bootstrap';
import { allTasks } from './state/options';

const handleAnswer = () => {
  const selectedCheckboxes =
Array.from(document.querySelectorAll('.active.carousel-item
input[type="checkbox"]:checked'));
  const selectedCheckboxesIds = selectedCheckboxes.map((checkbox) => checkbox.id);
  const activeTaskId = document.querySelector('.active.carousel-item h3').id;
  const currentTask = allTasks.find((task) => task.id === activeTaskId);
  const numberOfTrues = currentTask.options.filter((option) =>
option.isTrue).length;

  const wrongCheckboxes = currentTask.options.filter((option) => {
    if(selectedCheckboxesIds.includes(option.optionValue) && !option.isTrue) {
      return option;
    }
  })

  document.querySelectorAll('.active.carousel-item label').forEach((task) => {
    if(wrongCheckboxes.map((wrongTask) =>
wrongTask.optionValue).includes(task.htmlFor)) {
      console.log(task);
      task.style.color = 'red';
    };
  });

  if(wrongCheckboxes.length || numberOfTrues != selectedCheckboxesIds.length) {
    alert('Answer is false');
    return;
  }

  return alert('Answer is true');
}

function App() {
  return (
    <div className="App">
      <div className='title-page'>
        <h2>Training apparatus</h2>
        <div className = "sub-title">from the topic</div>
        <h2>Verification of programs</h2>
        <div className = "sub-title">disciplines</div>
        <h2>Theory of programming</h2>
        <a href="#slider">Start training</a>
        <div className = "author-info">
```

```
          <span>Author: </span><p>Сімперович М.М.</p> <br />
          <span>Lead: </span><p>Черненко О.О.</p>
        </div>
        <p style={{'margin-top': '20px'}} >ВНЗ Укоопспілки "Полтавський університет
економіки і торгівлі"</p>
        <p>ПОЛТАВА-2022</p>
      </div>
      <Carousel id="slider" variant="dark" interval={null}>
        {allTasks.map((task) => {
          return (
            <Carousel.Item id = {task.title}>
            <Carousel.Caption>
              <h3 style={{'margin-bottom': '120px'}} id =
{task.id}>{task.title}</h3>
              <fieldset style={{'margin-bottom': '120px'}}>
                {task.options.map((option) => {
                  return (<div key={option.optionValue} className = "anwer-
wrapper">
                    <label
htmlFor={option.optionValue}>{option.optionValue}</label>
                    <input type="checkbox" name={option.optionValue}
id={option.optionValue}/>
                  </div> )
                  })}
              </fieldset>
              <Button style={{'margin-bottom': '20px'}}
onClick={handleAnswer}>Відповісти</Button>
              <span>Wrong checked checkbox will be colored in red</span>
            </Carousel.Caption>
          </Carousel.Item>
          )
        })}
      </Carousel>
    </div>
  );
}

export default App;
```

ДОДАТОК В

PACKAGE.JSON

```
{
  "name": "interface",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "bootstrap": "^5.2.3",
    "eslint": "^8.29.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.6.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```