ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(*підпис*)

«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ ТРЕНАЖЕРА ДЛЯ СЛІПОГО ДРУКУ»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Рибак Віталій Андрійович
______ «___»_____2023 р.
(підпис)

Науковий керівник к.ф.-м.н., доцент, Черненко Оксана Олексіївна
_____ «___»_____2023 р.
(підпис)

Рецензент

ПОЛТАВА 2023 р.

ЗМІСТ

ВСТУП

У зв'язку з бурхливим розвитком інформаційних технологій, у багатьох користувачів персональних комп'ютерів виникла потреба швидко вводити текстову інформацію використовуючи комп'ютерну клавіатуру. Це й послугувало появі різноманітних програм-тренажерів для отримання та вдосконалення навичок набору тексту.

Клавіатурний тренажер – це вид комп’ютерних програм або онлайн-сервісів, призначених для навчання набору на комп’ютерній клавіатурі.

Зазвичай цілями тренажерів є:

- навчити сліпому методу друку, зокрема задіяти для набору всі десть пальців рук;
- збільшити швидкість набору;
- зменшити кількість помилок;
- поліпшити ритмічність набору (що дозволить зменшити втому при наборі).

Метою роботи є створення програмного забезпечення тренажеру з теми «Алгоритмізація та програмування тренажера для сліпого друку».

Об’єктом розробки є процес навчання сліпого друку.

Предметом розробки є програмний продукт, що реалізує тренажер з теми «Алгоритмізація та програмування елементів тренажера для сліпого друку».

Робота складається з чотирьох розділів. У першому розділі розглянуто постановку задачі. У другому розділі описано актуальність клавіатурних тренажерів, описано приклади існуючих тренажерів та історія створення клавіатури. В третьому розділі представлено алгоритмізацію задачі за темою роботи, обґрунтування вибору програмних засобів. В четвертому розділі описано розробку блок-схеми, процес програмної реалізації, роботу тренажеру.

Перелік використаних методів – застосування клавіатурних тренажерів та їх необхідність у сучасному світі, універсальна платформа для розміщення веб-проектів у вільному доступі для користувача GitHub Pages, мова програмування JavaScript.

Обсяг пояснювальної записки: 45 стор., в т.ч. основна частина – 34 стор., джерела – 15 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Для досягнення поставленої мети у роботі необхідно вирішити такі основні завдання:

- аналіз вимог і формальна постановка задачі;
- вибір або розробка математичної моделі;
- аналіз способів вирішення;
- логічне проектування і розробка алгоритму;
- кодування (написання програми);
- тестування і налагодження програмного забезпечення;
- впровадження, використання і супровід програмного забезпечення.

Перший і найважливіший етап у проектуванні програмного забезпечення - це постановка завдання розробки. Саме на цьому етапі створюється основа для майбутньої програми. Якщо на цьому етапі виникнуть помилки або будуть пропущені важливі деталі, це негативно позначиться на кінцевому результаті, і внесення змін і виправлень до такого проекту стане дуже проблематичним.

Першим кроком у проектуванні програмного забезпечення є формулювання цілей реалізації програми. На кожному етапі розробки чи впровадження можуть виникати непередбачені раніше проблеми та проект може вимагати певних змін, тому цей процес має бути гнучким і довгостроковим. Водночас для спрощення розробки слід заборонити різкі зміни вимог на етапі кодування (впровадження) програми. Необхідно також заздалегідь визначити умови внесення несуттєвих змін в проект. Всі домовленості такого плану повинні оформлюватися розробником і замовником програмного продукту в офіційному порядку у вигляді окремих пунктів у договорі на розробку програмного забезпечення, додаткових протоколів чи актів.

З офіційним висновком договору зазвичай передують:

- з'ясування реальної необхідності такої системи;
- оцінка можливості її розробки і зразкового обсягу витрат;
- визначення очікуваного ефекту від впровадження.

Після завершення попереднього дослідження буде створено список вимог до програмного забезпечення. Сюди входять:

- аналіз обстановки (сукупність умов, в яких передбачається експлуатувати програмну систему);
- опис виконуваних програмою системою функцій (чіткий опис того, що повинна робити система, на підставі яких вхідних даних, які дані є вихідними);
- обмеження, які повинні враховуватися в процесі проектування (терміни завершення; ресурси, наявні в наявності).

На стадіях формального аналізу проблеми та розробки вимог до програми головна мета полягає в тому, щоб знайти та представити ситуації, які можуть призвести до збою в роботі програми, а також пояснити, чому та як такі ситуації можна подолати.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Актуальність теми

Світ веб-розробки постійно змінюється. Досить важливим є уважно стежити за всім, що відбувається: актуальність існуючих фреймворків, вибір оптимального серед них, тенденції розробки та оформлення сайтів та інші.

Розробнику важливо бути в курсі поточних технологічних розробок, наскільки це можливо. Він повинен дивитися на можливості, але також враховувати можливі ризики, пов’язані з новими технологіями. Регулярні експерименти з різними структурами допомагають відчути, як потенційні вигоди, так і обмеження.

У зв’язку з високим попитом на онлайн-курси на просторах мережі Інтернет створюється велика кількість онлайн-тренажерів та посібників. Перевага даних ресурсів полягає в тому, що сам користувач вирішує, коли йому зручно витрачати час на виконання того чи іншого завдання. Онлайн-курси слідують такому принципу як дотримання часових рамок здачі завдання. У свою чергу, онлайн-тренажер/посібник не містять обмежень у часі.

На сьогоднішній день існує велика кількість тренажерів різного формату, серед яких наступні:

- тренажери для сліпого введення;
- тренажери для вивчення правил дорожнього руху;
- тренажери для вивчення іноземних мов;
- тренажери для вивчення мов програмування.

Тому майбутній онлайн-тренажер має бути корисним інструментом для людей, які хочуть вивчити навички друку або покращити свої навички. Особливо корисним цей інструмент може бути для дітей, або для дорослих, які хочуть покращити свої навички для роботи з документами.

Окрім того, тренажер для друку може бути корисним для людей з певними фізичними обмеженнями, такими як артрит, травми рук або інші моторні проблеми,

які забороняють виконувати повсякденні завдання, такі як друкування, звичайним способом.

Тренажер для друку дозволяє користувачам більш ефективно вивчати техніку друкування та підвищувати швидкість та точність.

Отже, тренажер для друку може бути корисним інструментом для навчання та покращення навичок друкування, особливо для людей з фізичними обмеженнями, які не можуть виконувати ці завдання звичайним способом.

2.2 Історія виникнення комп'ютерної клавіатури

Клавіатура комп'ютера – пристрій для введення інформації в комп'ютер та подачі сигналів керування. Містить стандартний набір клавіш друкарської машинки та деякі додаткові клавіші – керуючі та функціональні клавіші, клавіші керування курсором та малу цифрову клавіатуру.

Історія сучасної комп'ютерної клавіатури бере свій початок у 19 столітті з винаходом простої друкарської машинки. В 1868 Крістофер Латам Шольз (Christopher Sholes) запатентував свою першу розкладку як сукупність символів, що розташовувалися в алфавітному порядку. Подібна розкладка виявилася незручною, оскільки символи, що не часто використовуються, знаходилися на найпомітніших місцях і навпаки.

Вже в 1890 році була розроблена розкладка для друкарських машинок «QWERTY» (назва розкладки «QWERTY» походить від перших шести латинських літер на клавіатурі, починаючи від лівого верхнього кута зліва направо), яка до сьогоднішнього дня використовується при наборі тексту латинськими літерами. Кириличну розкладку «ЙЦУКЕН» створили так само в Америці наприкінці 19 століття. З того часу вона не зазнала сильних змін.

Ключовим моментом перетворення друкарської машинки на комп'ютерну клавіатуру став винахід наприкінці 19 століття теледрукарської машини Бодо. Цей

метод замінив телеграф, в якому інформація кодувалася двобітовим методом («точка-тире», а пізніше «наявність сигналу – відсутність сигналу»), а як пристрій введення використовувалися кнопки, що розташовуються в розкладці «QWERTY». 1943 ознаменувався появою комп'ютера ENIAC, який справив фурор у світі науки. Цей комп'ютер використовувався військовим для балістичних розрахунків. Вихідні дані він отримував за допомогою перфокарт і телетайпних стрічок, але програмне управління операціями здійснювалося за допомогою наборних панелей друкарських машин, що відповідають стандартам. 1960 рік є переломним моментом історія розвитку комп'ютерних клавіатур – ринку виходить електрична друкарська машинка. Вона мала ємнісну клавіатуру. Ємнісна клавіатура виготовлялася на друкованих текстолітових платах. Назва технології говорить сама за себе - цей тип працює за рахунок конденсаторів, розташованих у пристрої. Два майданчики з олова та нікельованої міді, які, до речі, ніяк не з'єднані один з одним, ні механічно, ні електрично формують кожну клавішу.

Клавіша натискається - відстань між майданчиками збільшується, при цьому змінюється електрична ємність, яка при натиснутому стані дорівнює приблизно 2 піко фарада, а при не натиснутому - 20 пико фарада. Зменшення ємності створює потік заряджених частинок, що обробляється контролером клавіатури, який у свою чергу генерує код натиснутої клавіші. Не дивлячись на складність конструкції, її головним плюсом стала легкість введення тексту для кінцевого користувача.

Наступний етап пов'язаний з розвитком розрахованих на багато користувачів систем, поділом машинного часу і появою терміналів. ЕОМ оснащувалися графічними дисплеями та телетайпами. Тепер користувачі могли бачити, який текст вони набирають, і навіть мали можливість його відразу редагувати.

Наприкінці 1970-х та на початку 1980-х років виробники стали випускати комп'ютери, які могли собі дозволити не лише офіси великих компаній, а й середній клас суспільства для домашнього використання. Комп'ютери являли собою клавіатуру, в яку був вбудований сам комп'ютер (процесор, ПЗУ, ОЗП та кілька модулів для додаткових пристроїв). Все це було виконано компактно та не займало багато місця. Клавіатури цих машин за функціями і за кількістю кнопок

перевершували електричні друкарські машини. Наприклад, було додано ряд функціональних клавіш Ctrl, Alt. Клавіша Enter виконувала функцію як повернення каретки, а й завершення введення даних. Для роботи з електронними документами були додані стрілки керування курсором, які поки що знаходилися на цифрах. Ці елементи знайшли своє активне застосування в графічних інтерфейсах та комп'ютерних іграх, які активно почали розвиватися, як тільки персональні комп'ютери набули широкого поширення.

Останній етап пов'язаний із розвитком модульних ПК, які мали можливість вдосконалення. Клавіатура стала окремим пристроєм, який підключався до комп'ютера за допомогою шнура і виглядав так: 83 клавіші, розділені на два блоки. Перший блок - алфавітно-цифровий, на ньому також розташовувалися стрілки управління, і другий - службовий (для системних клавіш). Основним недоліком було розміщення елементів керування. Незручне розташування функціональних клавіш, які виділялися розмірами проти іншими, послужило приводом до новаторських вдосконалень усталених стандартів.

Змінили розташування клавіш, їх кількість збільшили і нова клавіатура набула вже звичного сучасного вигляду.

За своїм призначенням клавіші на клавіатурі поділяються на шість груп:

1) функціональні;
2) алфавітно-цифрові;
3) управління курсором;
4) цифрова панель;
5) спеціалізовані;
6) модифікатори.

Дванадцять функціональних клавіш розташовані у верхньому ряду клавіатури. Нижче розташовується блок алфавітно-цифрових кнопок. Правіше цього блоку знаходяться клавіші керування курсором, а у правого краю клавіатури - цифрова панель.

До алфавітно-цифрового блоку належать клавіші для введення букв, цифр, знаків пунктуації та арифметичних дій, спеціальних символів. У стандартній

клавіатурі PC/AT цей блок містить 47 клавіш. Для деяких мов, де число літер в алфавіті більше 26, виробники клавіатур випускають клавіатури з додатковими клавішами в алфавітно-цифровому блоці - наприклад, на клавіатурах для української мови (33 літери) їх вже 48.

Клавіші алфавітно-цифрового блоку діляться рядами і зонами. Нижній ряд блоку знаходиться над клавішею «пробіл» та клавішами-модифікаторами Ctrl, Alt, AltGr. Він вважається першим. Вище – другий, у методі сліпого десяти пальцевого друку також званий «домашнім» поряд. Ще вище – третій. Найвищий ряд клавіш блоку - четвертий - у латинській розкладці QWERTY не містить клавіш для введення літер, але включає всі клавіші введення цифр. Тому його часто називають цифровим рядом.

Результат дії алфавітно-цифрових клавіш залежить від регістру (нижній або верхній) та рівня (перший чи другий), в якому здійснюється натискання цих клавіш.

На верхній частині клавіатури, а іноді в іншому місці розташовується блок так званих функціональних клавіш - від F1 до F12. Функції цих клавіш визначаються програмою та операційною системою, з якою користувач працює зараз. Часто програми встановлюють ті чи інші функції для комбінацій функціональних клавіш з клавішами Shift, Ctrl і Alt . У багатьох програмах при натисканні F1 на екрані виводиться вбудований довідник по програмі (часто вже відкритий на сторінці, що відповідає режиму програми, в якому вона знаходиться).

Основне призначення клавіш цифрової панелі – дублювання функцій клавіш алфавітно-цифрового блоку в частині введення цифр та арифметичних операторів. Клавіші панелі зручніші для введення цифр і арифметичних операторів, ніж клавіші алфавітно-цифрового блоку.

QWERTY - найбільш популярна латинська розкладка клавіатури, що використовується для англійської мови. На її основі створено більшість розкладок для мов, які використовують латиницю. Назва походить від 6 лівих символів верхнього ряду розкладки.

QWERTY - це спроба прискорити друк при тодішніх технічних обмеженнях - сліпий друк ще не винайдено, важелі зачіпають один за одного, а механізм друку знаходиться знизу і надрукований текст можна побачити, тільки провернувши барабан.

Але є й альтернативні розкладки. Розкладка QWERTY на даний момент критикується як анахронізм: проблема, яка призвела до появи розкладки QWERTY - літери, що зачіпають один за одного, - давно вирішена. З того часу було зроблено кілька спроб розробити розкладки, що найбільше підходять для швидкісного набору.

У 1936 році професор Вашингтонського університету Август Дворак вирішив повернутися до витоків стандартної розкладки та науково обґрунтувати необхідність нової. Результатом його досліджень стала нова розкладка, яка носить ім'я автора. Її принцип – максимальна зручність для набираючого. Але, незважаючи на те, що розкладка Двір розроблений за всіма правилами, враховує більшість міркувань ергономіки і включений до списку розкладок будь-якої версії Windows, з нею працюють лише 2% користувачів комп'ютерів.

Розкладка Colemak розроблена у 2006 році Шаєм Коулманом (Shai Coleman) для набору англійських текстів. Назва походить від Coleman + Dvorak. Розкладка пристосована до сучасних комп'ютерних реалій.

Набагато швидше за QWERTY і трохи швидше за Дворак - в Colemak розвантажені мізинці і частіше застосовується чергування рук.

Завдяки частковій схожості QWERTY та Colemak можна друкувати, наприклад, на QWERTY на роботі та на Colemak вдома. Розклад Дворака зовсім не схожий на QWERTY. Залишає важливі клавіатурні команди (Ctrl+Z, Ctrl+S тощо) у місцях, до яких можна дотягнутися однією рукою. Програмування на Colemak простіше, ніж на розкладці Дворака - всі розділові знаки, крім точки з комою, залишені на тій же позиції, що і в QWERTY. Цікаво, що на місці Caps Lock Colemak знаходиться ще одна клавіша ← Backspace і не тільки в цій розкладці різні клавіші [5].

2.2 Огляд існуючих клавіатурних тренажерів

Працюючи з текстом важлива швидкість його набору і особливо цей показник актуальний будь-яких операторів ЕОМ: програмістів, журналістів, секретарів – тобто, для тих хто постійно працює із великим потоком тексту. Клавіатурний тренажер — вид комп'ютерних програм чи онлайн-сервісів, призначених для навчання набору на комп'ютерній клавіатурі.

Цілями тренажерів є:

1) збільшення швидкості набору;

2) зменшення кількості друкарських помилок;

3) покращення ритмічності набору (що дозволяє зменшити втому при наборі).

При частому використанні програм «клавіатурних тренажерів» набір тексту відбувається вже рефлекторно, на підсвідомому рівні, коли запам'ятовуються розташування окремих літер і зручні комбінації їх поєднання. Постійні тренування є єдиним способом опанувати навички швидкого набору тексту з мінімальними друкарськими помилками.

Умовно, програми-клавіатурні тренажери поділяються на дві категорії. До першої відносяться програми, призначені лише для вдосконалення навичок набору тексту. Такі тренажери викликатимуть проблеми освоєння у новачків, проте вони будуть корисні решті, тим, хто постійно самоудосконалює свої вміння. До другого класу слід зарахувати тренажери, розроблені спеціально для навчання "з нуля". Як правило, вони побудовані у вигляді гри та націлені на те, щоб зацікавити користувача, створити почуття комфорту та зручності, з ускладненням завдань.

Розглянемо далі найвідоміші клавіатурні тренажери.

Keyboard Simulator. У цьому тренажері можна вибирати рівень своєї майстерності (від новачка до професіонала), змінювати розкладку клавіатури та мову інтерфейсу. При роботі ваше завдання – встигнути натиснути на клавіатурі ту літеру, яка падає. При цьому у верхній частині екрана ведеться проміжна

статистика, що складається з відсотка попадання та помилок. Остаточний результат виводиться окремим пунктом меню у вигляді діаграми швидкості друку та діаграми помилок.

Після знайомства вам запропонують вибрати один із трьох рівнів майстерності, далі розкажуть про правильну постановку пальців, а потім самі вправи.

Під рівнем майстерності мається на увазі певний рівень ритмічності, можливість допустити обмежену кількість помилок та швидкість друку. Робота над кожною вправою триватиме доти, доки ви не виконаєте ці вимоги.

Програма здатна запам'ятати останнє завдання, з яким користувачу вдалося впоратися, та статистику успішності (поточну та минулого разу).

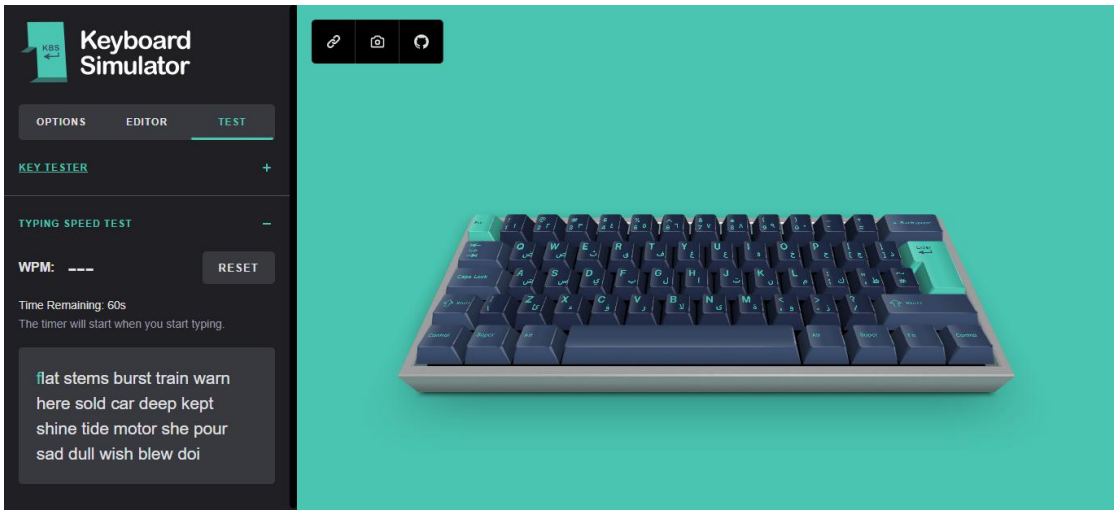


Рис. 2.1 – Екранна форма програми «Keyboard Simulator»

Stamina. Цей тренажер рекомендується людям, які досягли певних успіхів на терені швидкісного набирання тексту. Початківцям він буде не такий корисний, як інші подібні програми.

Програма Stamina є екраном, розділеним посередині горизонтальною смугою. До речі, Олексій Казанцев, автор цієї програми, запропонував нове розташування рук, вважаючи його більш прогресивним і природним: пальці лівої руки лежать на клавішах І, В, А , М, а правої - Т, О, Л, Д. Що ж, все залежить від анатомічних особливостей кожної людини.

Плюсом Stamina є те, що вона навчає друкуванню двома мовами: російською та англійською. Також вона має багато музичних файлів, які відтворюються під час роботи з програмою. Наприклад, якщо ви друкуєте рядок, що рухається, без помилок, то звучить приємна мелодія. Але варто розслабитися і натиснути не на ту літеру, як одразу "за кадром" лунає строгий кашель тощо.

Мінуси програми полягають у тому, що при закінченні роботи не видається статистика кількості набраних символів, швидкості роботи та відсотка допущених помилок.

Рис. 2.2 – Екранна форма програми «Stamina»

Соло на клавіатурі. Навчання у програмі побудовано у жартівливій формі: кожне завдання супроводжується веселими анекдотами, забавними фразами та не менш цікавими та водночас складними завданнями. Перед початком занять пропонується пройти курс, в якому вам розкажуть про правильну постановку рук на клавіатурі. Після цього йде відпрацювання набору букв, буквосполучень, слів і фраз.

Програма веде дуже докладну статистику занять, включаючи оцінку за повністю виконане завдання. Програмний продукт поширюється як умовно безкоштовний. Тому навіть під час випробувального терміну в тренажері доступна лише частина вправ (приблизно кілька десятків), під час виконання яких ви освоїте лише середній ряд.

До мінусів програми можна віднести те, що вона поширюється як умовно безкоштовна. Тому навіть під час випробувального терміну в тренажері доступна

лише частина вправ, під час виконання яких можна опанувати лише середній ряд клавіатури [6].

Рис. 2.3 – Екранна форма програми «Соло на клавіатурі»

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Алгоритмізація задачі за темою роботи

Крок 0. Користувачу відображається головна сторінка тренажера, на якій відображається назва тренажера, список доступних мов, кнопка для зміни стилю сайту (зміна кольору сторінки та інших елементів сайту), рівнів важкості та можливість вставити власний текст для тренування, також міститься інформація, щодо результатів набору тексту, а саме: кількість набраних користувачем символів у хвилину, слів у хвилину, та здійснених помилок. Нижче, відображається поле для вводу тексту та відображається сам текст для друкування, далі розташовується візуальна копія механічної клавіатури, на якій підсвічується натиснута поточна клавіша для візуального розуміння, яка клавіша була натиснута.

Крок 1. Користувачу для початку необхідно вибрати із списку мову, рівень важкості або вставити власний текст. Для вибору представлені дві мови: Українська, Англійська, та 4 рівні важкості для кожної мови. Після вибору пункту із запропонованого списку з’явиться текст для тренування, та відбувається перехід на крок 2.

Крок 2. Користувачу необхідно курсором миші навести на поле для введення символів та натиснути ліву кнопку миші, інакше він не зможе розпочати тренування.

Крок 3. Почати друкувати текст, що відображається на екрані. При натисканні користувачем першого символу починається відлік поточного рядка для обрахунку надрукованих символів у хвилину, слів у хвилину та обрахунку відсотку помилок в даному рядку. Якщо користувач натиснув символ, що не співпадає із поточним символом у тексті то даний символ засвічується жовтим кольором у всьому рядку, що вказує бути уважнішим при натисканні клавіши, що відповідає даному символ.

Крок 4. Коли користувач закінчить друкувати, йому буде виведено повідомлення «You have successfully completed! In order to start again, press the

Refresh Page button.», що буде свідчити про завершення тренування, та буде запропоновано розпочати знову, тоді відбувається перехід до кроку 5.

Крок 5. Якщо користувач вирішив продовжити тренування, йому необхідно натиснути відповідну кнопку «Refresh Page», що обновить тренажер, і дозволить розпочати тренування або вибрати інший пункт із списку запропонованого тренування.

Крок 6. Коли користувач вибрав пункт – Вставити текст, тоді на сайті зникне візуальна копія клавіатури та рядок для введення, і з’явиться поле для введення власного тексту. Коли користувач вставив (завершив вводити власний текст) то йому необхідно натиснути кнопку «Продовжити», після чого знову з’явиться візуальна клавіатура, та рядок для введення тексту, але якщо поле в яке користувачу необхідно ввести власний текст є пустим то кнопка не спрацює, необхідно ввести хоча б 1 символ.

Крок 7. Коли користувач завершив тренування, і бажає розпочати знову, тоді йому необхідно виконати Крок 5.

3.2 Обґрунтування вибору програмних засобів для реалізації завдання роботи

Для кваліфікаційної роботи обрано середовище розробки Microsoft Visual Studio 2013, мову програмування JavaScript. Visual Studio 2013 − один з продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інструментальних засобів, також мову розмітки HTML та мову стилів CSS.

JavaScript – це мова програмування, що дозволяє зробити Web -сторінку інтерактивною, тобто такою що реагує на дії користувача**.**

Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим в звичайний Web -браузер. Іншими словами, код програми вбудовується в HTML - документ і виконується на

боці клієнта. Для виконання програми не потрібно навіть перезавантажувати Web - сторінку, всі програми виконуються у відповідь на будь-яку подію. Наприклад, перед відправленням даних форми можна перевірити їх на допустимі значення і, якщо значення не відповідають очікуваним, заборонити відправлення даних.

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об’єктів додатків. Найбільш широке застосування знаходить у браузерах, як мова сценаріїв для надання інтерактивності веб-сторінкам.

Основні архітектурні риси:

- динамічна типізація,
- автоматичне керування пам'яттю,
- прототипне програмування,
- функції як об'єкти першого класу.

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожою на Java, але при цьому легкою для використання непрограмістами. Також JavaScript має низку властивостей об’єктно-орієнтованої мови, але реалізоване в мові прототипування обумовлює відмінності в роботі з об’єктами в порівнянні з традиційними об’єктно-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мова, - функції як об’єкти першого класу, об’єкти як списки, каррінг, анонімні функції, замикання – що додає мові додаткову гнучкість.

Незважаючи на схожий з Сі синтаксис, JavaScript у порівнянні з мовою Сі має корінні відмінності:

- об’єкти, з можливістю першого класу;
- функції як об’єкти першого класу;
- автоматичне приведення типів;
- автоматичне прибирання сміття;
- анонімні функції.

У мові відсутні такі корисні речі, як:

- модульна система – JavaScript не надає можливості управляти залежностями та ізоляцією областей видимості;
- стандартна бібліотека – зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управління потоками вводу/виводу, базових типів для бінарних даних;
- стандартні інтерфейси до веб-серверів та баз даних;
- система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

Синтаксис мови JavaScript дуже нагадує синтаксис Сі і Java, семантично ж мова набагато ближче до Selfi, Smalltalk або навіть Ліспу.

Структурно JavaScript можна представити у вигляді об’єднання трьох частин, що чітко різняться одна від одної:

- ядро (ECMAScript);
- об’єктна модель браузера (Browser Object Model або BOM)
- об'єктна модель документа (Document Object Model або DOM).

Об'єктну модель документа іноді розглядають як окрему від JavaScript сутність, що узгоджується з визначенням DOM як незалежного від мови інтерфейсу документа.

Одна з популярних технологій, що дозволила зробити сторінки динамічнішими і забезпечити нові можливості — це динамічне завантаження і вставка даних в документ, що отримала назву AJAX [7].

HTML (Hypertext Markup Language) - це стандартна мова розмітки для створення веб-сторінок. HTML використовується для створення структури і контенту веб-сторінок, тобто він дозволяє визначати, які елементи відображатимуться на веб-сторінці і як вони будуть виглядати.

HTML з'явився у 1990 році і був створений Тімом Бернерсом-Лі, британським вченим, який працював у ЦЕРНі (Європейський організація з ядерних досліджень). Початково, HTML був створений як проста мова розмітки для наукових документів, щоб легко обмінюватися інформацією між науковцями з усього світу.

HTML складається з різних елементів, таких як заголовки, параграфи, списки, таблиці, зображення, посилання тощо. Кожен елемент має свій властивий синтаксис, атрибути та значення, які вказують, як цей елемент має бути відображений на сторінці. Крім того, HTML також дозволяє використовувати CSS (Cascading Style Sheets) для оформлення веб-сторінок та JavaScript для додавання функціональності. Використання цих технологій дозволяє створювати більш складні та інтерактивні веб-сторінки.[8]

CSS (Cascading Style Sheets) - це мова стилів, яка використовується для опису вигляду веб-сторінок, написаних мовою розмітки HTML. З допомогою CSS можна задавати такі параметри, як колір тексту, фону, шрифту, розташування елементів на сторінці та інше.

CSS був розроблений як доповнення до HTML, який не мав можливості для опису вигляду сторінки. Перша версія CSS була випущена в 1996 році і мала досить обмежений набір можливостей, але вже тоді вона відкрила безліч можливостей для веб-дизайнерів та розробників, які змогли здійснити більш гнучкий та виразний дизайн веб-сторінок. З часом CSS стала все більш популярною та потужною мовою стилів, і разом з виходом нових версій стандарту стала включати все більше можливостей, що дозволяли створювати більш складні та привабливі веб-сайти. Сьогодні CSS3 є останньою версією стандарту CSS і містить велику кількість нових функцій та можливостей, таких як анімації, трансформації та гнучкий дизайн для різних пристроїв.

Один з головних принципів CSS - розділення змісту веб-сторінки від її візуального оформлення. Це дозволяє зберігати сторінки відповідними до стандартів та зробити їх більш доступними для людей з обмеженими можливостями.

CSS дозволяє відокремити стиль веб-сайту від HTML-коду, що дозволяє зменшити код сторінки, зберегти зусилля при редагуванні та підтримці веб-сайту, а також забезпечити його швидке завантаження.[9]

4. ПРАКТИЧНА ЧАСТИНА

4.1. Розробка блок-схеми, яка підлягає програмуванню

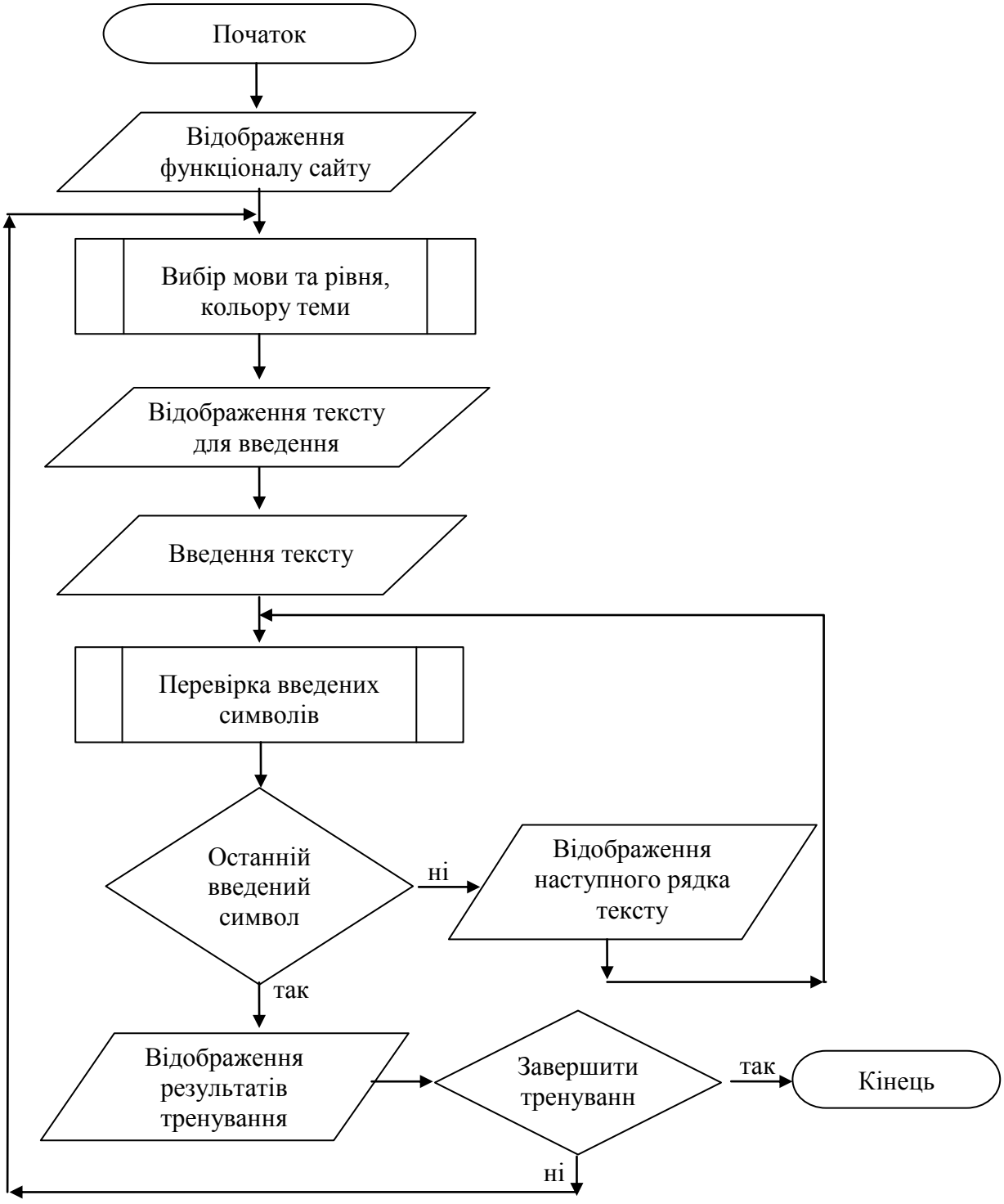
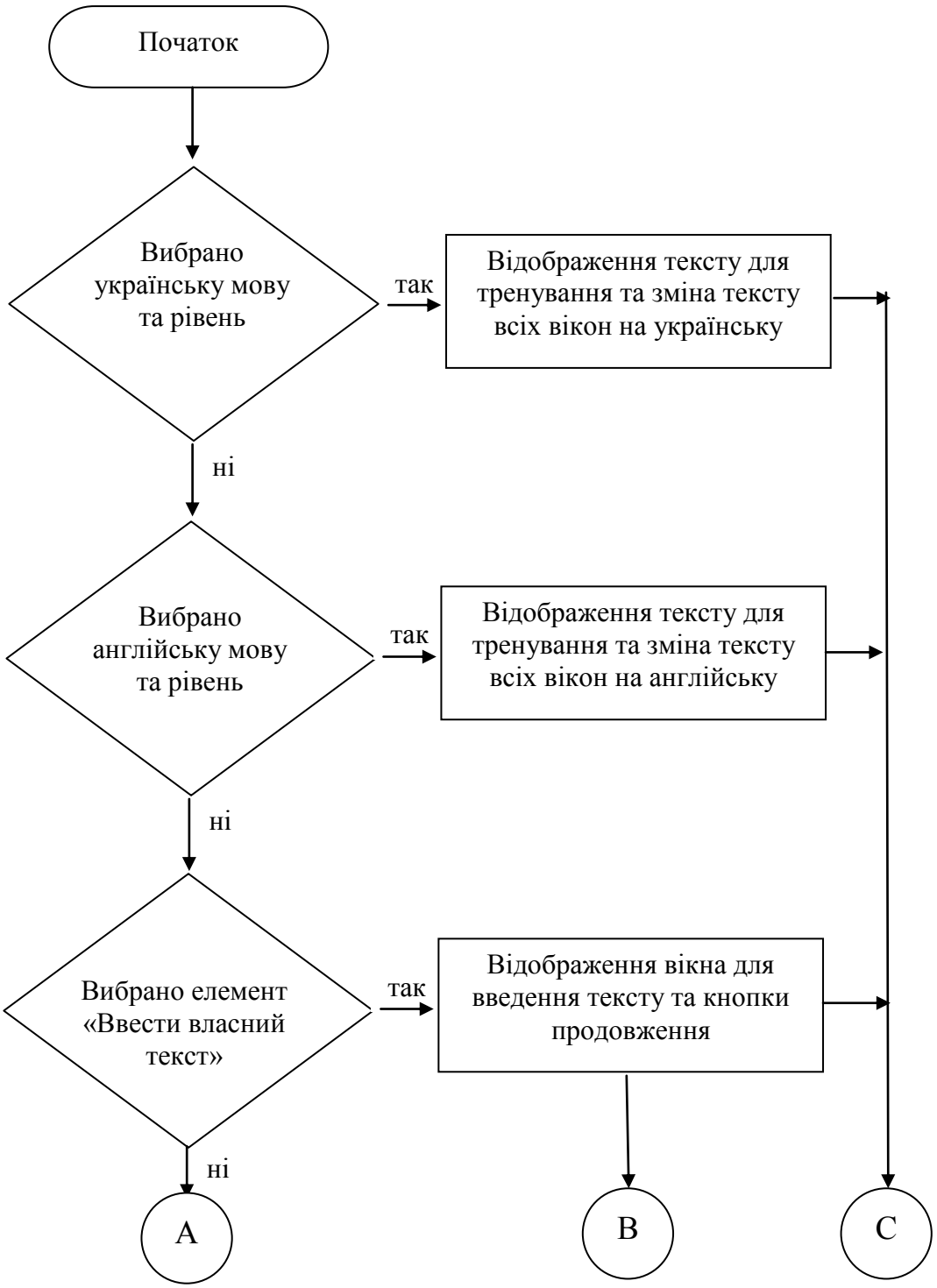


Рисунок 4.1 – Блок-схема роботи тренажеру


Початок
Вибрано українську мову та рівень
так
Відображення тексту для тренування та зміна тексту всіх вікон на українську
ні
Вибрано англійську мову та рівень
так
Відображення тексту для тренування та зміна тексту всіх вікон на англійську
ні
Вибрано елемент «Ввести власний текст»
так
Відображення вікна для введення тексту та кнопки продовження
ні
A
B
C

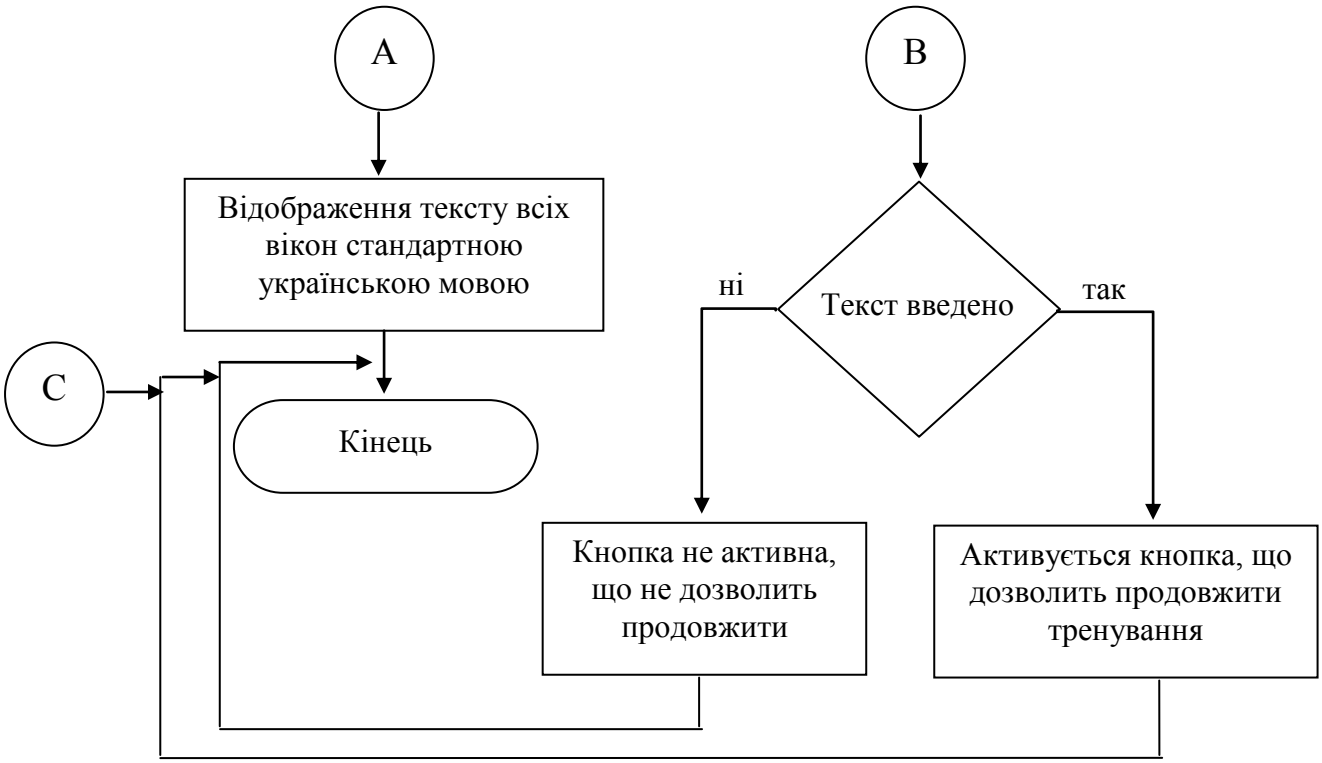


Рисунок 4.2 – Блок-схема вибору мови та рівня важкості

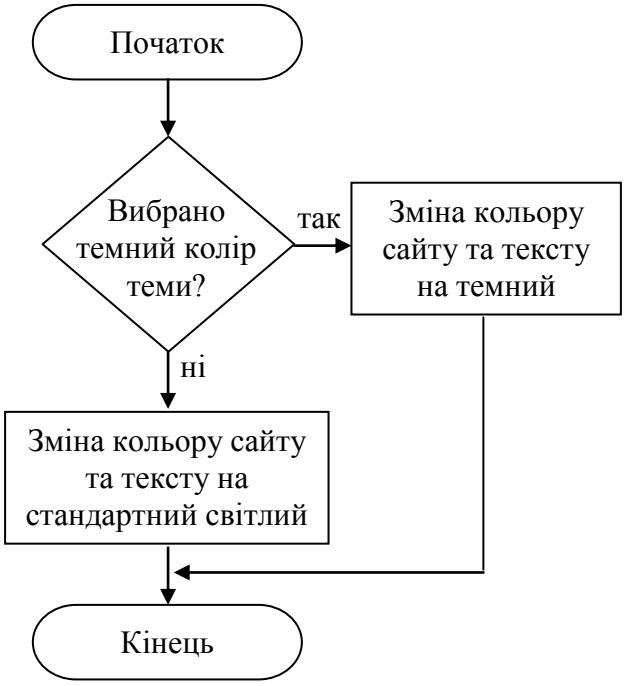


Рисунок 4.3 – Блок-схема вибору кольору теми сайту

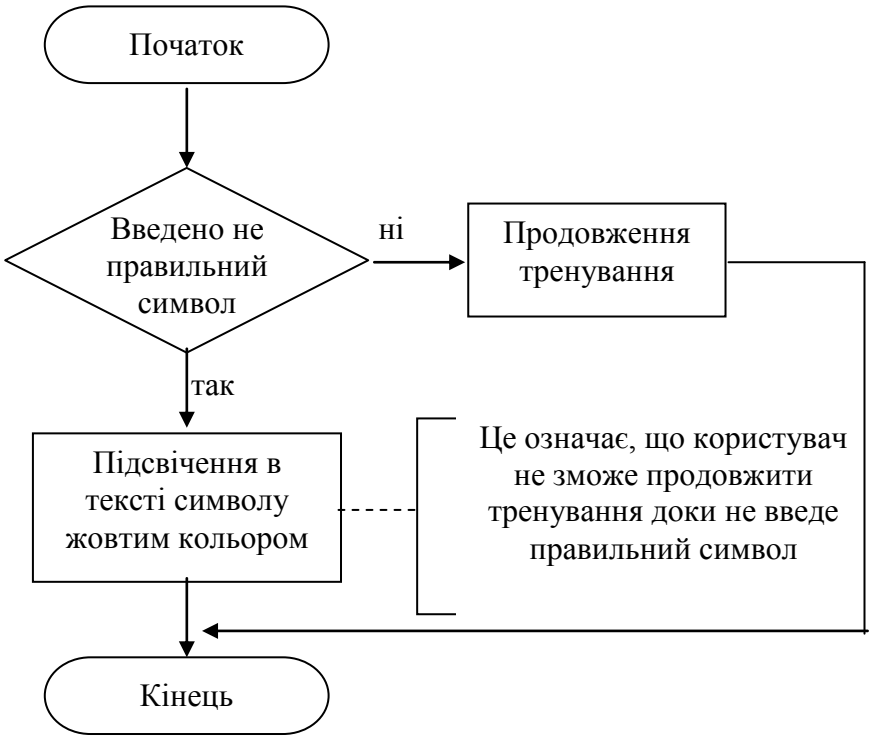


Рисунок 4.4 – Блок-схема перевірки введених символів

4.2 Опис процесу програмної реалізації

Навчальний тренажер на тему “Тренажер для сліпого друку” розміщено на платформі GitHub Pages. GitHub Pages – це безкоштовний сервіс хостингу веб-сторінок, який надається компанією GitHub.

Він дозволяє розробникам захистити та опублікувати статичні веб-сайти з використанням Git та GitHub. За допомогою GitHub Pages, розробники можуть розміщувати свої веб-сайти в обліковому записі GitHub та публікувати їх у веб-просторі. Це забезпечує простий та швидкий спосіб розгортання веб-сайту без необхідності використовувати власний хостинг.

Для створення тренажеру було обрано мову програмування JavaScript, вона є скриптовою мовою, що виконується на стороні клієнта (в браузері) та на стороні сервера (за допомогою Node.js). Вона дозволяє додавати динамічний функціонал до веб-сторінок, такий як анімація, валідація форм, маніпулювання DOM-елементами, взаємодія з користувачем.

Відображення списку із доступними мовами та рівнями:

```
<div class="navigation">
        <div class="block-select">
          <select title="Вибір мови/Language selection" id="select">
            <option value="0" selected="true" disabled="disabled">Вибір мови|Language selection</option>
            <option disabled>Українська</option>
            <option value="1_y">1 рівень</option>
            <option value="2_y">2 рівень</option>
            <option value="3_y">3 рівень</option>
            <option value="4_y">4 рівень</option>
            <option value="Українська">тренування</option>
            <option value="-" disabled>----------</option>
            <option disabled>English</option>
            <option value="1_e">1 level</option>
            <option value="2_e">2 level</option>
            <option value="3_e">3 level</option>
            <option value="4_e">4 level</option>
            <option value="English">training</option>
            <option value="-" disabled>----------</option>
            <option id="optionText" value="text">Вставити текст|Insert text</option>
          </select>
          <button value="Refresh Page" onClick="window.location.reload();" title="Refresh Page">
            <img src="img/Refresh.svg" alt="Refresh Page" width="19px" height="19px">
          </button>
        </div>
        <div class="block-theme">
```

```
            <button class="darck-t theme">Darck theme</button>
            <button class="light-t theme">Light theme</button>
          </div>
        </div>
```

Виведення інформації про помилки користувача підчас тренування:

```
<div class="results">
    <div class="results-title  r_s_t-1">Результати попередніх рядків:</div>
    <div class="results-title  r_s_t-2">The results of the previous rows:</div>

    <div class="indicators">
      <div class="indicator">
        <div class="indicator__icon">
          <i class="fas fa-tachometer-alt"></i>
        </div>
        <div class="indicator__name r_s_t-1">Символів у хвилину:</div>
        <div class="indicator__name r_s_t-2">Symbols per minute:</div>
        <div class="indicator__value" id="symbolsPerMinute">0</div>
      </div>
      <div class="indicator">
        <div class="indicator__icon">
          <i class="far fa-clock"></i>
        </div>
        <div class="indicator__name r_s_t-1">Слів у хвилину:</div>
        <div class="indicator__name r_s_t-2">Words per minute:</div>
        <div class="indicator__value" id="wordsPerMinute">0</div>
      </div>

      <div class="indicator">
        <div class="indicator__icon indicator__icon--errors">
```

```
                <i class="fas fa-exclamation-circle"></i>
            </div>
            <div class="indicator__name r_s_t-1">Помилок:</div>
            <div class="indicator__name r_s_t-2">Errors:</div>
            <div class="indicator__value" id="errorPercent">0%</div>
          </div>
        </div>
      </div>
    </div>
```

Виведення повідомлення про успішне завершення тренування:

```
<div class="text textFinish show">
    <span>You have successfully completed!
        In order to start again, press the Refresh Page button.</span>
</div>
```

Якщо користувач вирішить ввести власний текст для тренування, то на екрані з'явиться поле введення тексту та кнопка для продовження:

```
<div class="container texterea text-hide">
  <div>
    <textarea name="text" id="text" cols="95" rows="15"  placeholder="Введіть текст|Insert text"></textarea>
    <button type = "button" class="clr" id="submit" name = "button">
      <span class="r_s_t-1">Продовжити</span>
      <span class="r_s_t-2">Continue</span>
    </button>
  </div>
</div>
```

4.3 Опис роботи тренажеру

Тренажер складається з одної сторінки на якій відображується весь функціонал (рис. 4.5):

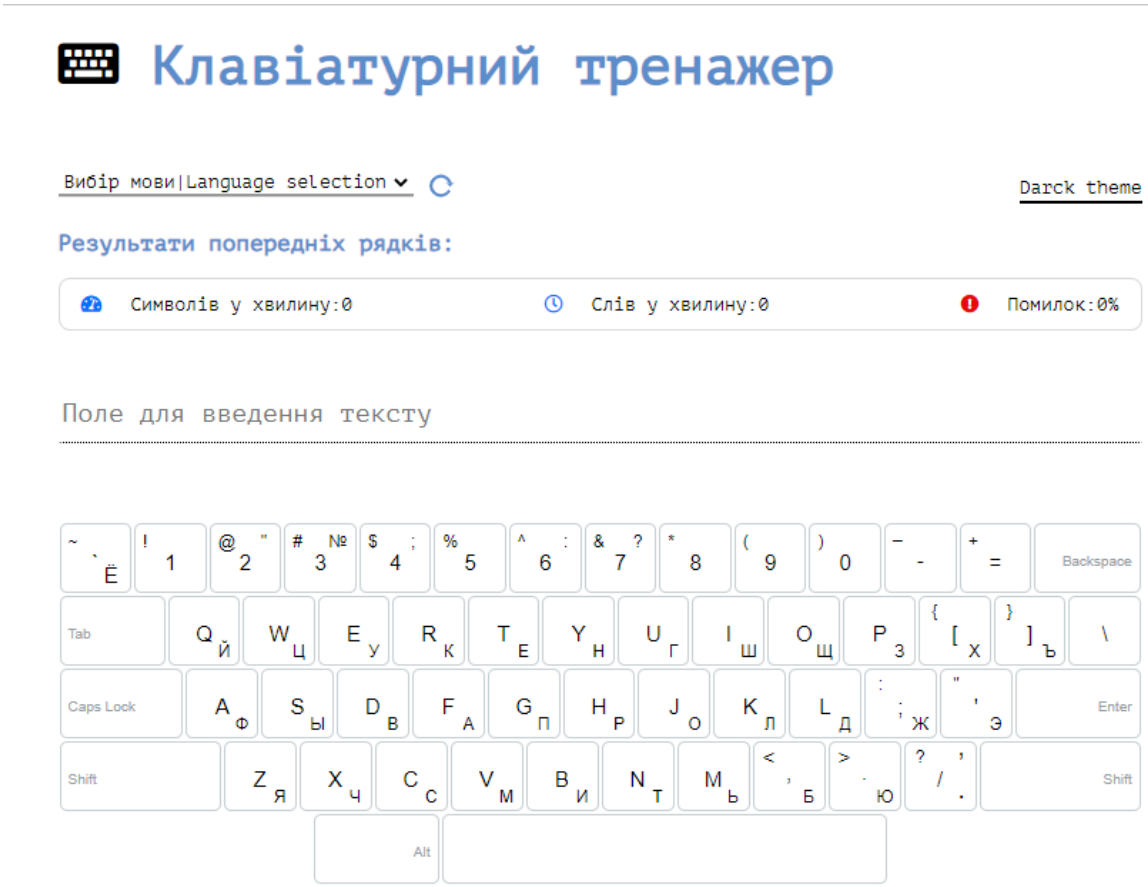


Рисунок 4.5 – Основна сторінка

Зміна кольору теми здійснюється натисканням на кнопку `Darck theme` (рис. 4.6):

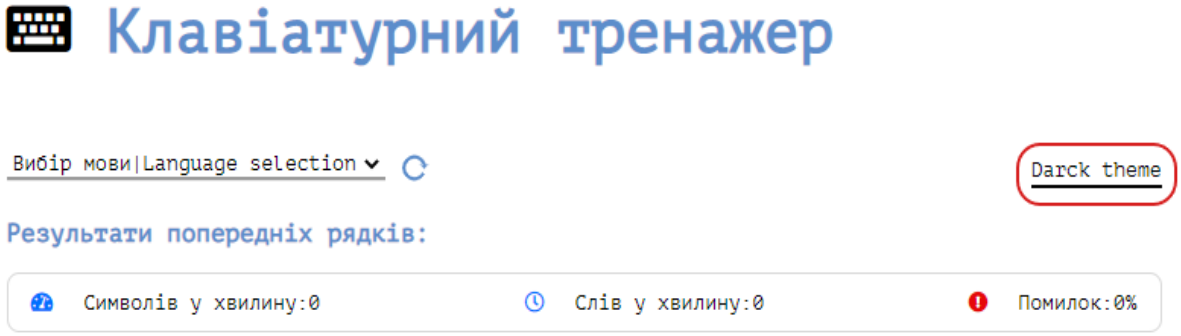


Рисунок 4.6 – Зміна кольору теми

Вибір рівня важкості та мови відбувається по кліку на випадаючий список “Вибір мови | Language selection”. У даному списку доступні 2 мови на вибір, а саме: Українська та Англійська, а до кожної мови доступні 4 рівні важкості. Також доступний пункт, де можна вставити власний текст (рис.4.7):

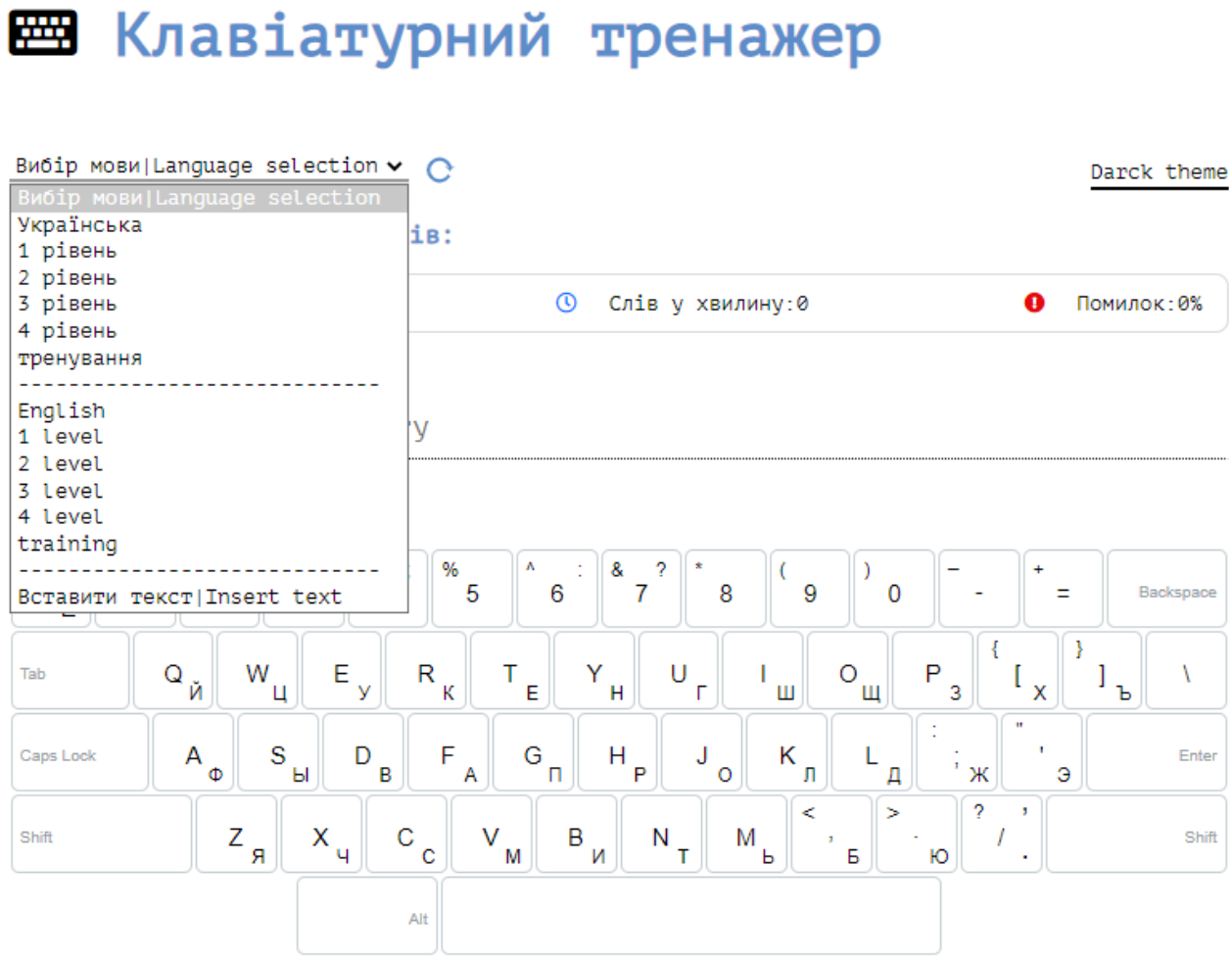


Рисунок 4.7 – Вибір рівня важкості та мови тренування

Вибравши пункт “Вставити текст | Insert text”, користувачу буде доступне поле для введення власного тексту та кнопка (рис. 4.8):

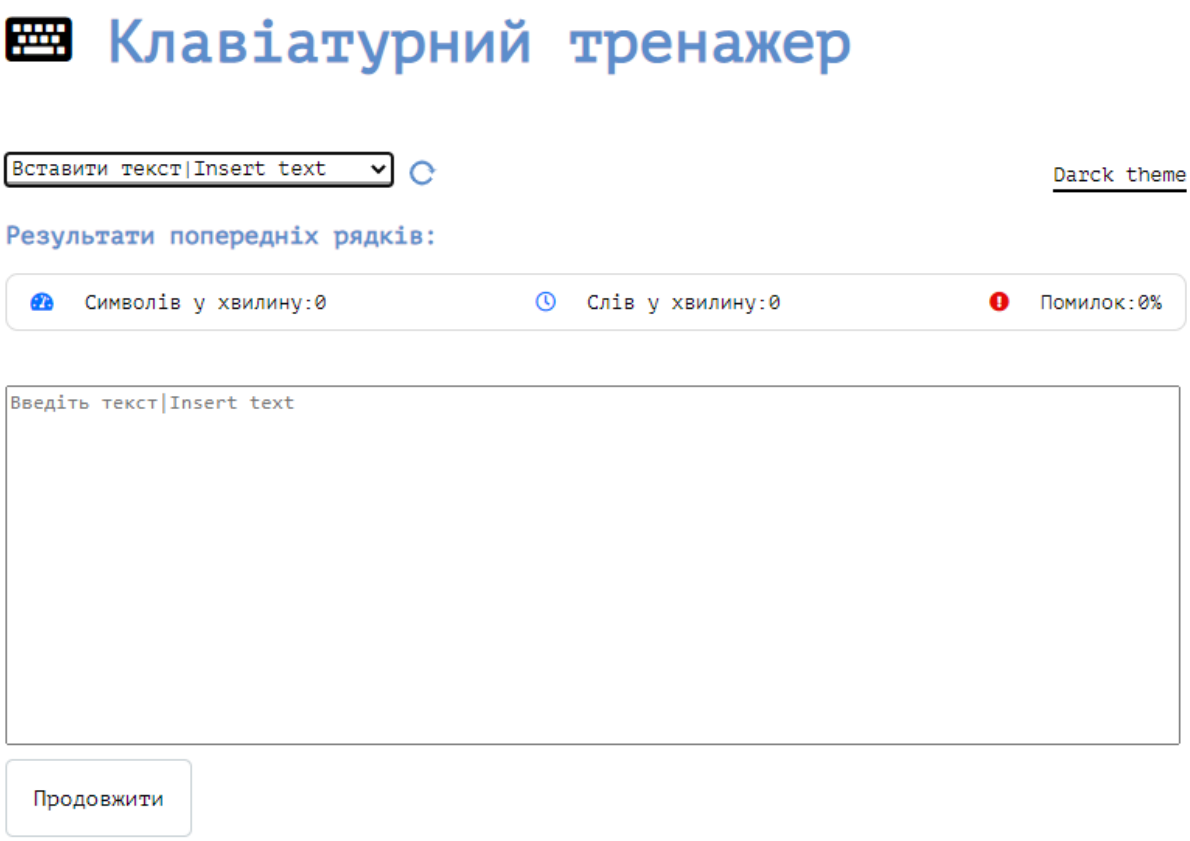


Рисунок 4.8 – Відображення “Вставити текст | Insert text”

Якщо користувач не увів текст і вирішить продовжити, тоді буде виведено повідомлення про те, що необхідно ввести текст (рис. 4.9).

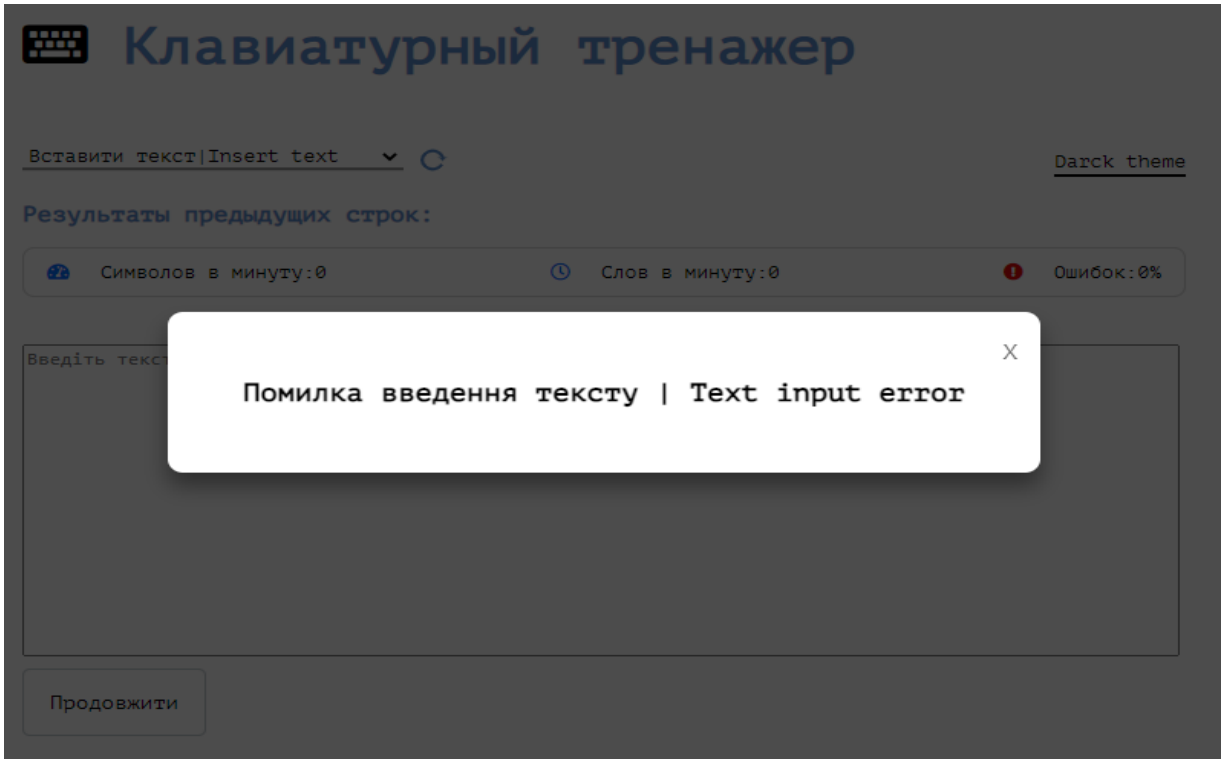


Рисунок 4.9 – Відображення повідомлення про помилку введення тексту

Коли користувач розпочне тренування йому буде виведено текст для друку. Також, якщо користувач ввів не правильний символ, то йому це буде відображатись підсвічуванням символу в тексті - жовтим кольором у всьому тексті. Продовжити можна буде тільки тоді, коли буде введено вірний символ (рис. 4.10).

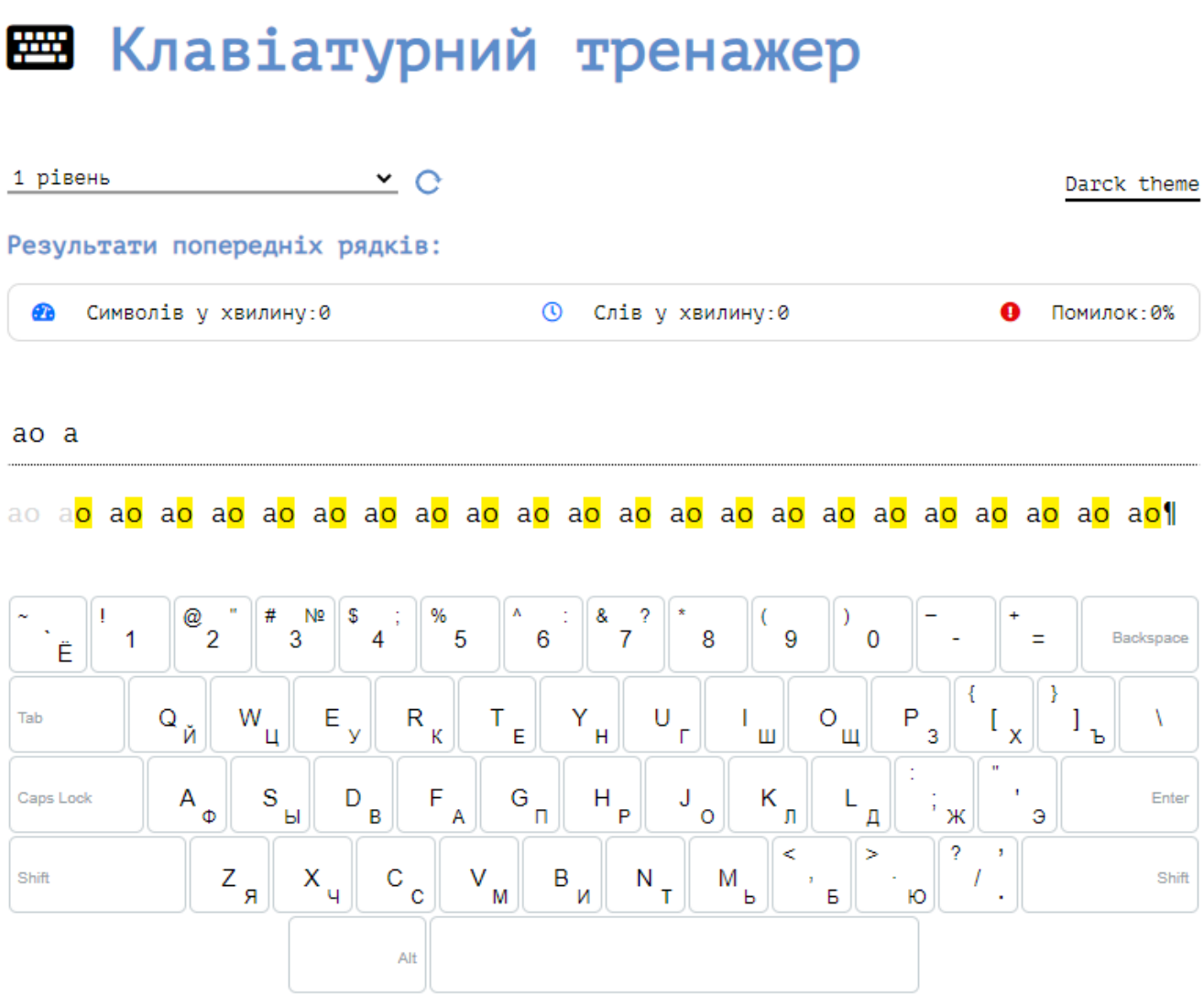


Рисунок 4.10 – Відображення тексту та неправильно введеного символу.

Після того, як користувач завершить проходження певного рівня важкості, йому буде виведено повідомлення про завершення тренування та результат його тренування (рис. 4.11).

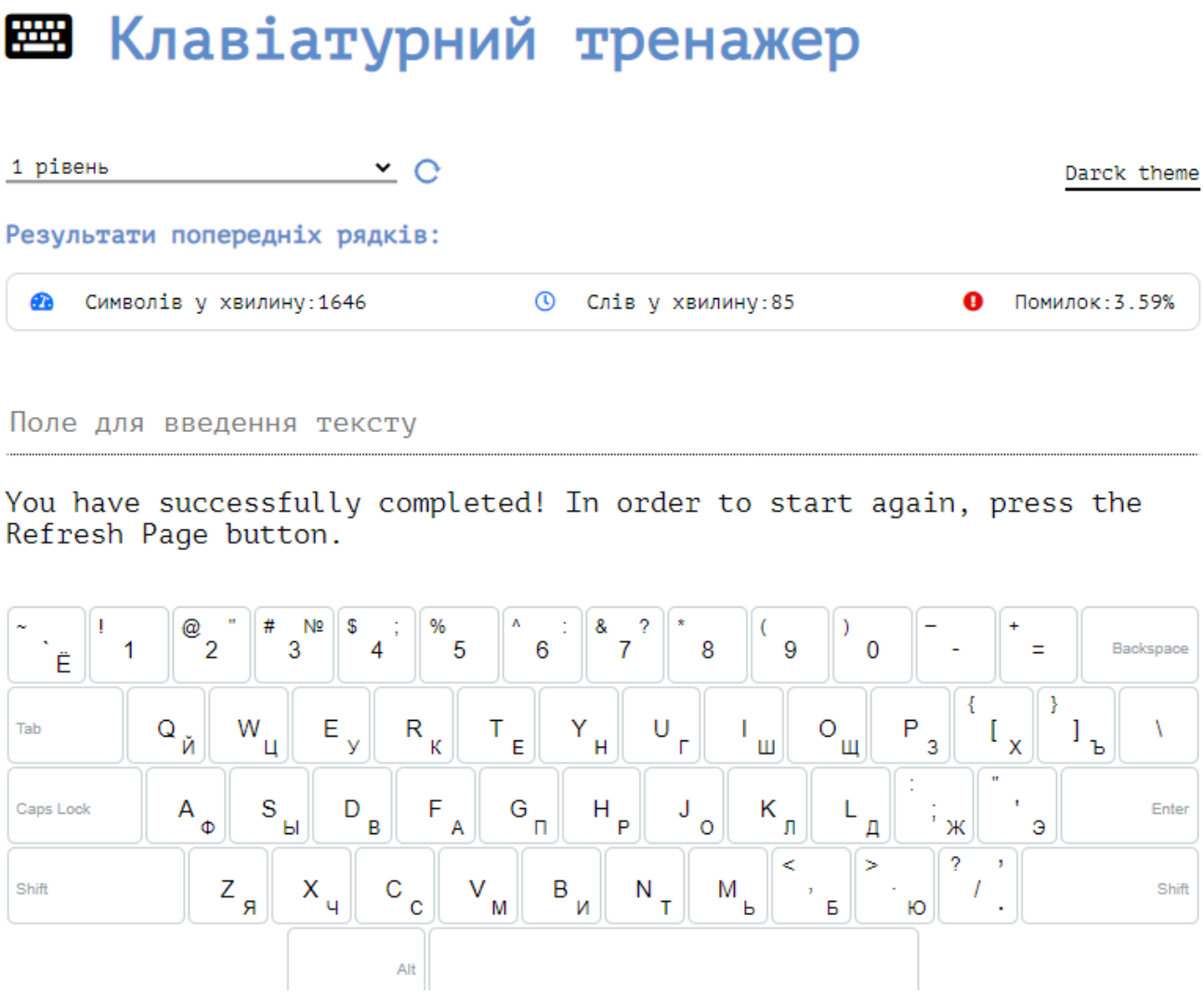


Рисунок 4.11 – Після завершення тренування користувач отримає повідомлення про завершення та побачить свої результати тренування.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є алгоритм програмного продукту, що реалізує тренажер з теми «Алгоритмізація та програмування елементів тренажера для сліпого друку».

Розглянемо виконані завдання проекту:

- описано постановку задачі;
- описано інформаційний огляд;
- переглянутий теоретичний матеріал з теми;
- розроблено алгоритм тренажеру.

В алгоритмі користувача описано процес тестування:

- вибір кольору теми сайту;
- вибір рівнів важкості та мови;
- встановлення послідовності кроків.

При неправильно введеному символі підчас тренування відображається помилка.

Завершивши тренування виводиться відповідне повідомлення: «You have successfully completed! In order to start again, press the Refresh Page button.».

Підчас виконання кваліфікаційної роботи було використано мову програмування JavaScript, мову розмітки HTML та спеціальну мову стилів CSS.

Основні результати роботи:

- сформульовані основні вимоги до тренажера, що розроблявся;
- проведено огляд плюсів та мінусі існуючих тренажерів;
- розглянуто доступні у мережі програми-тренажери;
- складено алгоритм роботи із створеним тренажером.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Ольховська О.В. Методичні рекомендації до виконання кваліфікаційної роботи для студентів за освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» / О.В. Ольховська, О.О. Черненко. – Полтава: ПУЕТ, 2022. – 71 с.
2. Стандарт вищої освіти підготовки бакалаврів спеціальності 122 «Комп'ютерні науки» (наказ МОН України № 962 від 10.07.2019 р.).
3. Освітньо-професійна програма «Комп'ютерні науки». Ступінь – бакалавр / Полтавський університет економіки та торгівлі. 2022. – 23 с.
4. Адам Д. Скотт. Розробка на JavaScript, 2021р. – 20 с.
5. Клавіатурні тренажери як спосіб самостійного оволодіння технікою швидкого друку [Електронний ресурс] – Режим доступу: https://vseosvita.ua/library/klaviaturni-trenazeri-ak-sposib-samostijnogo-ovolodinna-tehnikou-svidkogo-druku-83463.html.
6. Історія створення комп'ютерної клавіатури [Електронний ресурс] – Режим доступу: https://isoft.kz/obzori/rozhdenie_klaviaturi.html.
7. Огляд клавіатурних тренажерів для навчання методу сліпого друку [Електронний ресурс] – Режим доступу: https://jak.koshachek.com/articles/ogljad-klaviaturnih-trenazheriv-dlja-navchannja.html.
8. Опис мови програмування та її можливостей [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti.
9. HTML: мова розмітки гіпертексту [Електронний ресурс] – Режим доступу: https://webdoky.org/uk/docs/Web/HTML/#kliuchovi-resursy.
10. Джон Дакетт. HTML і CSS. Розробка і Дизайн Веб-сайтів, 2011р. – 213с.
11. Джон Дакетт. JavaScript і jQuery. Розробка і дизайн веб-сайтів, 2017р.
12. Елізабет Росон, Ерік Фрімен. Вивчаємо програмування на JavaScript, 2014р.

13. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.
14. Бен Фрейн. Адаптивний дизайн на HTML5 та CSS3 для будь-яких пристроїв, 2022р. – 121с.
15. Ніколас Закас. Професійний JavaScript для веб-розробників, 2012р. – 201с.

ДОДАТОК А. Програма

Відображення копії фізичної клавіатури для візуального сприйняття:

```
<div class="keyboard">
            <div class="line">
               <div id="letter_ë" data-letters="Ëë`~"><p class="p_1">~</p>`<p
class="p_2">Ë</p></div>
               <div id="letter_1" data-letters="1!"><p class="p_1">!</p>1 </div>
               <div id="letter_2" data-letters='2@'"><p class="p_1">@</p>2<p class="p_3
p_32">"</p></div>
               <div id="letter_3" data-letters="3#№"><p class="p_1">#</p>3<p class="p_3
p_33">№</p></div>
               <div id="letter_4" data-letters="4$;"><p class="p_1">$</p>4<p class="p_3
p_34">;</p></div>
               <div id="letter_5" data-letters="5%"><p class="p_1">%</p>5</div>
               <div id="letter_6" data-letters="6^:"><p class="p_1">^</p>6<p class="p_3
p_35">:</p></div>
               <div id="letter_7" data-letters="7&?"><p class="p_1">&</p>7<p class="p_3
p_36">?</p></div>
               <div id="letter_8" data-letters="8*"><p class="p_1">*</p>8</div>
               <div id="letter_9" data-letters="9("><p class="p_1">(</p>9</div>
               <div id="letter_0" data-letters="0)"><p class="p_1">)</p>0</div>
               <div id="letter_minus" data-letters="_-–"><p class="p_1 p_1_">_</p>-
</div>
               <div id="letter_plus" data-letters="+="><p class="p_1">+</p>=</div>
               <div class="backspace" id="letter_backspace" data-spec="backspace">
                  Backspace
               </div>
            </div>
            <div class="line">
               <div class="tab" id="letter_tab" data-spec="tab">Tab</div>
               <div id="letter_й" data-letters="ЙйQq">Q <p class="p_4">Й</p></div>
               <div id="letter_ц" data-letters="ЦцWw">W <p class="p_4">Ц</p></div>
               <div id="letter_у" data-letters="УуEe">E <p class="p_4">У</p></div>
               <div id="letter_к" data-letters="КкRr">R <p class="p_4">К</p></div>
               <div id="letter_е" data-letters="ЕеTt">T <p class="p_4">Е</p></div>
               <div id="letter_н" data-letters="НнYy">Y <p class="p_4">Н</p></div>
               <div id="letter_г" data-letters="ГгUu">U <p class="p_4">Г</p></div>
               <div id="letter_ш" data-letters="ШшIi">I <p class="p_4">Ш</p></div>
               <div id="letter_щ" data-letters="ЩщOo">O <p class="p_4">Щ</p></div>
               <div id="letter_з" data-letters="ЗзPp">P <p class="p_4">З</p></div>
```

```
            <div id="letter_x" data-letters="Xx{["><p class="p_8">{</p> [ <p
class="p_4">Х</p></div>
            <div id="letter_ъ" data-letters="Ъъїї}]"><p class="p_8">}</p> ] <p
class="p_4">Ъ</p></div>
            <div id="letter_slash" data-letters="\">\</div>
         </div>
         <div class="line">
            <div class="caps" id="letter_capslock" data-spec="capslock">
               Caps Lock
            </div>
            <div id="letter_ф" data-letters="ФфAa">A <p class="p_4">Ф</p></div>
            <div id="letter_ы" data-letters="ЫыIiSs">S <p class="p_4">Ы</p></div>
            <div id="letter_в" data-letters="ВвDd">D <p class="p_4">В</p></div>
            <div id="letter_а" data-letters="АаFf">F <p class="p_4">А</p></div>
            <div id="letter_п" data-letters="ПпGg">G <p class="p_4">П</p></div>
            <div id="letter_р" data-letters="РрHh">H <p class="p_4">Р</p></div>
            <div id="letter_о" data-letters="ОоJj">J <p class="p_4">О</p></div>
            <div id="letter_л" data-letters="ЛлKk">K <p class="p_4">Л</p></div>
            <div id="letter_д" data-letters="ДдLl">L <p class="p_4">Д</p></div>
            <div id="letter_ж" data-letters="Жж:;"><p class="p_8">:</p> ; <p
class="p_4">Ж</p></div>
            <div id="letter_э" data-letters='Ээ Є є"&#39;' data-letters="""><p
class="p_8">"</p> ' <p class="p_4">Э</p></div>
            <div class="enter" id="letter_enter" data-spec="enter">Enter</div>
         </div>
         <div class="line">
            <div class="shift" data-spec="shift">Shift</div>
            <div id="letter_я" data-letters="ЯяZz">Z <p class="p_4">Я</p></div>
            <div id="letter_ч" data-letters="ЧчXx">X <p class="p_4">Ч</p></div>
            <div id="letter_с" data-letters="СсCc">C <p class="p_4">С</p></div>
            <div id="letter_м" data-letters="МмVv">V <p class="p_4">М</p></div>
            <div id="letter_и" data-letters="ИиBb">B <p class="p_4">И</p></div>
            <div id="letter_т" data-letters="ТтNn">N <p class="p_4">Т</p></div>
            <div id="letter_ь" data-letters="ЬьMm">M <p class="p_4">Ь</p></div>
            <div id="letter_б" data-letters="Бб,<"><p class="p_punct">,</p> <p
class="p_5"><</p> <p class="p_б">Б</p></div>
            <div id="letter_ю" data-letters="Юю.>"><p class="p_punct">.</p> <p
class="p_5">></p> <p class="p_ю">Ю</p></div>
            <div id="letter_dot" data-letters="?./"><p class="p_8">?</p> / <p
class="p_6">,</p> <p class="p_7">.</p></div>
            <div class="shift" data-spec="shift">Shift</div>
         </div>
         <div class="line">
            <div class="alt" id="letter_alt" data-spec="alt">Alt</div>
```

```
            <div class="space" id="letter_space" data-spec="space"></div>

          </div>
        </div>
        <!-- // Keyboard -->
      </div>
```

Виведення повідомлення про успішно завершене тренування:

```
<div class="text textFinish show">
    <span>You have successfully completed! In order to start again, press the Refresh Page button.</span>
 </div>
```

Приклад функціоналу правильності введення символу користувача, відображення тексту відповідно вибраного користувачем рівня важкості та обробка помилок:

```
init();

function init() {
  input.addEventListener("keydown", keydownHandler);
  input.addEventListener("keyup", keyupHandler);

  viewUpdate();
}

function keydownHandler(event) {
  event.preventDefault();
  const letter = letters.filter((x) => x.dataset.letters.includes(event.key));

  let key = event.key.toLowerCase();

  if (key === " ") {
    key = "space";
    press(" ");
  }

  if (key === "enter") {
    press("\n");
  }

  const ownSpecs = specs.filter((x) => x.dataset.spec === key);

  if (ownSpecs.length) {
```

```
        ownSpecs.forEach((spec) => spec.classList.add("pressed"));
        return;
    }

    console.warn("Не известный вид клавиши.", event);
    if (letter) {
        if (element.value === "Українська" || element.value === "Русский" || element.value
=== "1_y" || element.value === "2_y" || element.value === "3_y" || element.value ===
"4_y" || element.value === "1_p" || element.value === "2_p" || element.value === "3_p" ||
element.value === "4_p") {
            if(event.key === ',' || event.key === '.'){
                letter.forEach(function(spec, i) {
                    if (i==1) {
                        spec.classList.add("pressed");
                    } });
                press(event.key);
                return;
            }
            if(event.key === ';' || event.key === ':' || event.key === '?'){
                letter.forEach(function(spec, i) {
                    if (i==0) {
                        spec.classList.add("pressed");
                    } });
                press(event.key);
                return;
            }
        }
        if (element.value === "English" || element.value === "1_e" || element.value ===
"2_e" || element.value === "3_e" || element.value === "4_e"){
            if(event.key === ',' || event.key === '.'){
                letter.forEach(function(spec, i) {
                    if (i==0) {
                        spec.classList.add("pressed");
                    } } );
                press(event.key);
                return;
            }
            if(event.key === ';' || event.key === ':' || event.key === "" || event.key === '?'){
                letter.forEach(function(spec, i) {
                    if (i==1) {
                        spec.classList.add("pressed");
                    } });
                press(event.key);
                return;
```

```
            }
        }
        letter.forEach((spec) => spec.classList.add("pressed"));
        press(event.key);
        return;
    }
}

function keyupHandler(event) {
    event.preventDefault();

    const letter = letters.filter((x) => x.dataset.letters.includes(event.key));

    let key = event.key.toLowerCase();

    if (key === " ") {
        key = "space";
    }

    const ownSpecs = specs.filter((x) => x.dataset.spec === key);

    if (ownSpecs.length) {
        ownSpecs.forEach((spec) => spec.classList.remove("pressed"));
        return;
    }

    if (letter) {
        letter.forEach((spec) => spec.classList.remove("pressed"));

        return;
    }
}

function createParty(text) {
    const party = {
        text,
        strings: [],
        maxStringLength: 70,
        maxShowStrings: 1,
        currentStringIndex: 0,
        currentPressedIndex: 0,
        errors: [],
        started: false,
```

```
        statisticFlag: false,
        timerCounter: 0,
        startTimer: 0,
        errorCounter: 0,
        commonCounter: 0,
    };

    party.text = party.text.replace(/\n/g, "\n ");
    const words = party.text.split(" ");

    let string = [];
    for (const word of words) {
        const newStringLength =
            [...string, word].join(" ").length + !word.includes("\n");
        if (newStringLength > party.maxStringLength) {
            party.strings.push(string.join(" ") + " ");
            string = [];
        }

        string.push(word);

        if (word.includes("\n")) {
            party.strings.push(string.join(" "));
            string = [];
        }
    }

    if (string.length) {
        party.strings.push(string.join(" "));
    }

    return party;
}

function press(letter) {
    party.started = true;

    if (!party.statisticFlag) {
        party.statisticFlag = true;
        party.startTimer = Date.now();
    }

    const string = party.strings[party.currentStringIndex];
    const mustLetter = string[party.currentPressedIndex];
```

```
    let psl = party.strings.length-1;

    if (letter === mustLetter) {
        party.currentPressedIndex++;

        if (string.length <= party.currentPressedIndex) {
            party.currentPressedIndex = 0;
            if (party.currentStringIndex === psl) {
                const doneClass = document.querySelector('.text');
                const finishClass =document.querySelector('.textFinish');
                doneClass.classList.add('hide');
                finishClass.classList.remove('show');
            }else{
                party.currentStringIndex++;
            }
            party.statisticFlag = false;
            party.timerCounter = Date.now() - party.startTimer;
            party.errors.length=0;
        }
    } else if (!party.errors.includes(mustLetter)) {
        party.errors.push(mustLetter);
        party.errorCounter++;
    }
    party.commonCounter++;

    viewUpdate();
}

function viewUpdate() {
    const string = party.strings[party.currentStringIndex];
    const showedStrings = party.strings.slice(
        party.currentStringIndex,
        party.currentStringIndex + party.maxShowStrings
    );

    const div = document.createElement("div");
    const firstLine = document.createElement("div");
    firstLine.classList.add("lineS");
    div.append(firstLine);

    const done = document.createElement("span");
    done.classList.add("done");
    done.textContent = string.slice(0, party.currentPressedIndex);
```

```
firstLine.append(
    done,
    ...string
        .slice(party.currentPressedIndex)
        .split("")
        .map((letter) => {

            if (letter === "\n") {
                return "¶";
            }

            if (party.errors.includes(letter)) {
                const errorSpan = document.createElement("span");
                errorSpan.classList.add("hint");
                errorSpan.textContent = letter;
                return errorSpan;
            }

            return letter;
        })
);

for (let i = 1; i < showedStrings.length; i++) {
    const line = document.createElement("div");
    line.classList.add("lineS");
    div.append(line);

    line.append(
        ...showedStrings[i].split("").map((letter) => {
            // if (letter === " ") {
            //  return "·";
            // }

            if (letter === "\n") {
                return "¶";
            }

            return letter;
        })
    );
}
textExample.innerHTML = "";
textExample.append(div);
```

```
    input.value = string.slice(0, party.currentPressedIndex);

    const wordsMinute = string.match(/[^\s]+/g);

    if (!party.statisticFlag && party.started) {

        wordsPerMinute.textContent = Math.round(
            (60000 * wordsMinute.length) / party.timerCounter
        );

        symbolsPerMinute.textContent = Math.round(
            (60000 * party.commonCounter) / party.timerCounter
        );

        errorPercent.textContent = Math.floor((10000 * party.errorCounter) /
party.commonCounter) / 100 + "%";
    }
```