МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
ВІДДІЛЕННЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА
КОМП'ЮТЕРНІ НАУКИ»
ЦИКЛОВА КОМІСІЯ СПЕЦІАЛЬНОСТІ «КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи
освітньо-професійний ступінь «фаховий молодший бакалавр»
на тему:

«Розробка програми «Клавіатурний тренажер»

Виконав студент (ка) _4_ курсу, групи П-41
спеціальності 122 «Комп'ютерні науки»

Лісовський Микола Анатолійович

Керівник Устименко Ярослав Іванович

Рецензент _____

Житомир – 2024 року

Зміст

Вступ 5
РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ 7
1.1. Постановка основної задачі розробки _____ 7
1.2. Огляд та порівняння аналогічних систем. _____ 9
1.3. Вибір та обґрунтування технологій розробки _____ 11
Висновки до розділу 1 _____ 15
РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА 16
2.1. Вибір алгоритмів та їх ефективність _____ 16
2.2. Структура програми _____ 17
2.3. Програмна реалізація _____ 19
2.4. Опис класів _____ 20
2.5. Тестування програмного продукту _____ 21
Висновки до розділу 2 _____ 22
РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМНИМ ПРОДУКТОМ 23
3.1. Мінімальні вимоги до системи _____ 23
3.2. Інтерфейс користувача _____ 24
Висновки до розділу 3 _____ 26
ВИСНОВКИ 27
Перелік літературних джерел 28
Додаток А 31

РЕФЕРАТ

Записка: 40 стор., 4 рис., 1 додаток, 20 джерел.

Ключові слова: КЛАВІАТУРНИЙ ТРЕНАЖЕР, ПРОГРАМНА РОЗРОБКА, ШВИДКІСТЬ НАБОРУ ТЕКСТУ, ТОЧНІСТЬ НАБОРУ ТЕКСТУ, ІНТЕРФЕЙС КОРИСТУВАЧА, ТЕХНОЛОГІЇ РОЗРОБКИ, АНАЛІЗ АНАЛОГІЧНИХ СИСТЕМ, ВИМОГИ ДО СИСТЕМИ, ПРОГРАМНІ ВИМОГИ, ТЕСТУВАННЯ ПРОГРАМИ, РЕАЛІЗАЦІЯ АЛГОРИТМІВ, ОЦІНКА ЕФЕКТИВНОСТІ.

Дипломна робота на тему "Розробка програмного клавіатурного тренажера" присвячена створенню і реалізації програмного засобу, спрямованого на покращення швидкості та точності набору тексту на клавіатурі. У роботі розглянуто актуальність використання клавіатурних тренажерів, проаналізовано існуючі аналогічні системи та розроблено вимоги до системи. Було обрано та обґрунтовано технології розробки, проведено детальний огляд програмної реалізації та описано особливості роботи з інтерфейсом користувача. Висновки дипломної роботи вказують на важливість розробленого клавіатурного тренажера та його практичний інтерес для широкого кола користувачів.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

GUI - Графічний Інтерфейс Користувача.

WPF - Windows Presentation Foundation, технологія для створення графічних інтерфейсів в програмах під управлінням операційної системи Windows.

UI - Інтерфейс користувача.

IDE - Інтегроване середовище розробки.

API - Інтерфейс програмування застосунків.

ASCII - Американський стандарт коду для обміну інформацією.

URL - Уніфікований локаційний локатор, адреса в Інтернеті.

HTML - Мова розмітки гіпертексту.

CSS - Каскадні таблиці стилів.

.NET - Платформа розробки програмного забезпечення Microsoft.

Вступ

На даний момент в світі інформаційних технологій вміння швидко і точно вводити текст на комп'ютері стало необхідним навиком для більшості професій. Десятипальцевий метод сліпого набору, який раніше був привілеєм лише професійних машиністок і секретарів, сьогодні є важливим для всіх, хто регулярно працює з текстами на комп'ютері. Це включає підготовку рефератів, написання електронних листів, участь у форумах чи онлайн-конференціях. Швидкість і точність набору тексту можуть суттєво вплинути на продуктивність праці, а також на комфорт роботи за комп'ютером, зменшуючи навантаження на очі та руки.

Багато людей, навіть ті, хто часто працює за комп'ютером, досі використовують для набору тексту лише кілька пальців, що значно знижує їхню продуктивність і збільшує кількість помилок. Відсутність систематичного підходу до навчання навичкам сліпого набору може призводити до постійної втоми та дискомфорту під час роботи. На сучасному етапі розвитку інформаційних технологій існує нагальна потреба у створенні ефективних інструментів для навчання сліпому методу набору тексту, які допоможуть користувачам різного віку та професій значно покращити свої навички.

Метою даної дипломної роботи є розробка програми "Клавіатурний тренажер", яка забезпечить ефективне навчання навичкам сліпого набору тексту, покращить швидкість та точність друкування, а також зменшить втому користувачів під час роботи за комп'ютером.

Для досягнення поставленої мети необхідно вирішити такі завдання:

Проаналізувати існуючі методики навчання сліпому набору тексту та визначити їх переваги і недоліки.

Розробити концепцію програми, яка включатиме різноманітні вправи для тренування всіх пальців рук, а також інструменти для оцінки швидкості і точності набору тексту.

Реалізувати програму на платформі WPF, забезпечивши зручний і інтуїтивно зрозумілий інтерфейс користувача.

Провести тестування програми серед цільової аудиторії для оцінки її ефективності та зручності використання.

Внести корективи до програми на основі зібраних відгуків і результатів тестування.

Об'єктом дослідження є процес навчання сліпому методу набору тексту. Предметом дослідження є програмні засоби для навчання швидкому і точному набору тексту, зокрема програма "Клавіатурний тренажер".

Наукова новизна роботи полягає в розробці ефективного інструменту для навчання сліпому методу набору тексту, який поєднує в собі сучасні методики навчання та зручний інтерфейс користувача. Практичне значення роботи полягає в можливості широкого застосування розробленої програми серед користувачів різного віку та професій для покращення їхніх навичок роботи з текстами на комп'ютері.

Дипломна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі представлено аналіз існуючих методів навчання сліпому набору тексту. Другий розділ присвячений розробці концепції та реалізації програми "Клавіатурний тренажер". У третьому розділі викладено результати тестування програми та аналіз її ефективності.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Основною метою розробки програми "Клавіатурний тренажер" є створення ефективного інструменту для навчання швидкому і точному набору тексту сліпим методом. Ця програма повинна допомогти користувачам покращити свої навички друкування, зменшити кількість помилок та підвищити загальну продуктивність праці за комп'ютером.

Основні задачі:

- провести дослідження існуючих програм та методик навчання сліпому набору тексту;
- визначити сильні та слабкі сторони існуючих рішень;
- виявити потреби і вимоги користувачів, які можна врахувати під час розробки;
- визначити основні функціональні можливості програми, включаючи:

- набір вправ для тренування всіх пальців рук;
- інструменти для оцінки швидкості та точності набору тексту;
- розробити зручний і інтуїтивно зрозумілий інтерфейс користувача;
- вибрати відповідну технологічну платформу для розробки (WPF);
- реалізувати основні функції програми, такі як створення та виконання вправ, відстеження результатів, надання зворотного зв'язку;
- забезпечити стабільність і продуктивність роботи програми;
- провести тестування програми на різних етапах розробки для виявлення помилок і недоліків.

Розроблена програма "Клавіатурний тренажер" повинна забезпечити користувачам ефективний інструмент для навчання сліпому методу набору тексту, який допоможе, значно підвищити швидкість та точність друкування, зменшити кількість помилок під час набору тексту, зменшити втому очей та

рук під час тривалої роботи за комп'ютером, підвищити загальну продуктивність праці.

Виконання даної задачі вимагає ретельного планування, розробки та тестування програмного продукту. Розробка програми "Клавіатурний тренажер" стане важливим внеском у покращення навичок роботи з комп'ютером для широкого кола користувачів, сприяючи підвищенню їхньої професійної ефективності.

1.2. Огляд та порівняння аналогічних систем.

Аналогічні системи - це програми та онлайн-ресурси, які також призначені для навчання та тренування навичок друку на клавіатурі. Ось деякі характеристики аналогічних систем:

TypingClub - це онлайн-платформа, що пропонує безкоштовні уроки з друку на клавіатурі для усіх рівнів. Вона має інтерактивні вправи, відстеження прогресу та можливість налаштування швидкості та складності.

The Typing Cat: Ця платформа також пропонує безкоштовні уроки з друку на клавіатурі, а також ігри та вправи для покращення швидкості та точності печаті.

Keybr відома своїми інтерактивними вправами та груповими уроками з друку на клавіатурі. Вона надає різні методи тренування та аналізує результати, щоб допомогти користувачам покращити свої навички.

Rapid Typing - це безкоштовна програма для Windows, яка має велику кількість уроків та ігор для навчання друку на клавіатурі. Вона надає детальну статистику про прогрес користувача.

TypingTest.com спеціалізується на проведенні тестів на швидкість та точність друку. Вона пропонує різноманітні вправи та тести, які допомагають оцінити та покращити навички друку на клавіатурі.

Ці аналогічні системи також мають свої унікальні особливості та переваги, і вони можуть бути використані для досягнення схожих цілей з навчання та покращення друку на клавіатурі.

У процесі розробки програми "Клавіатурний тренажер" важливо врахувати існуючі аналоги на ринку, їхні переваги та недоліки. Це допоможе створити конкурентоспроможний продукт, який буде ефективно задовольняти потреби користувачів. Розглянемо кілька популярних програм для навчання сліпому методу набору тексту та порівняємо їхні ключові характеристики.

1. TypingMaster — це популярний комерційний продукт, призначений для навчання сліпому набору тексту. Програма пропонує різні уроки, вправи та ігри для тренування навичок друкування.

Переваги:

- Інтерактивні уроки з детальними інструкціями.
- Різноманітні вправи та ігри для підтримки інтересу користувачів.
- Вбудована система відстеження прогресу і надання зворотного зв'язку.
- Можливість персоналізації навчального процесу.

Недоліки:

- Комерційна ліцензія, що вимагає покупки для повного доступу до функцій.
- Інтерфейс може здаватися застарілим для сучасних користувачів.
- Відсутність інтеграції з іншими платформами і додатками.

2. Klavaro — це безкоштовний і відкритий клавіатурний тренажер, який підтримує різні розкладки клавіатури і надає базові функції для навчання сліпому набору.

Переваги:

- Безкоштовний і відкритий вихідний код.
- Підтримка різних розкладок клавіатури.
- Легкий у використанні інтерфейс.
- Можливість налаштування власних уроків і вправ.

Недоліки:

- Обмежені функції в порівнянні з комерційними продуктами.
- Відсутність інтерактивних елементів та ігор для підтримки інтересу користувачів.
- Простіший дизайн, який може не відповідати сучасним стандартам.

3. Keybr.com — це онлайн клавіатурний тренажер, який пропонує уроки з поступовим підвищенням складності, засновані на аналізі помилок користувача.

Переваги:

- Доступність онлайн, не вимагає встановлення.
- Інтерактивні уроки з аналізом помилок і наданням зворотного зв'язку.
- Система адаптивного навчання, яка підлаштовується під індивідуальні потреби користувача.

- Підтримка різних мов і розкладок клавіатури.

Недоліки:

- Необхідність постійного підключення до Інтернету.
- Обмежені функції в безкоштовній версії.
- Відсутність розширених вправ та ігор для підтримки інтересу користувачів.

4. Ratatype — це онлайн-платформа для навчання сліпому набору тексту, яка пропонує курси, тести та сертифікацію для користувачів.

Переваги:

- Доступність онлайн, без необхідності встановлення.
- Курси з поступовим підвищенням складності.
- Можливість отримання сертифікатів після проходження тестів.
- Підтримка спільноти користувачів для змагань і взаємодії.

Недоліки:

- Обмежені функції в безкоштовній версії.
- Необхідність постійного підключення до Інтернету.
- Можливі рекламні вставки в безкоштовній версії.

Аналіз аналогічних систем показує, що для створення конкурентоспроможної програми "Клавіатурний тренажер" необхідно поєднати найкращі риси існуючих рішень, такі як інтерктивність, адаптивне навчання, зручний інтерфейс та доступність онлайн. Важливо також врахувати потреби користувачів, забезпечивши гнучкість налаштувань і різноманітність навчальних матеріалів та ігор для підтримки мотивації. Ці елементи допоможуть створити ефективний і популярний інструмент для навчання сліпому методу набору тексту.**1.3. Вибір та обґрунтування технологій розробки**

При розробці програмного продукту "Клавіатурний тренажер" важливо вибрати відповідні технології, які забезпечать ефективність, надійність та зручність у використанні. Основні критерії для вибору технологій включають простоту розробки, можливість підтримки і розширення програми,

кросплатформенність та зручність користування. У цьому розділі буде охарактеризовано вибір технологій для розробки програми та обґрунтовано їх використання.

Мова програмування: C#

Вибір: Для розробки "Клавіатурного тренажера" обрана мова програмування C#.

Обґрунтування:

- Зручність розробки: C# — це високорівнева мова програмування, яка забезпечує простоту та швидкість розробки.
- Підтримка об'єктно-орієнтованого програмування (ООП): Мова C# підтримує принципи ООП, що сприяє написанню модульного, масштабованого та легкозрозумілого коду.
- Стабільність та продуктивність: C# є стабільною мовою з високою продуктивністю, що важливо для створення ефективних програм.
- Інтеграція з іншими технологіями Microsoft: C# тісно інтегрується з платформою .NET, що дозволяє використовувати всі можливості цієї платформи для розробки надійних додатків.

Платформа: .NET Framework / .NET Core

Вибір: Для розробки вибрано .NET Core (останню версію .NET), яка є сучасною кросплатформенною версією .NET Framework.

Обґрунтування:

- Кросплатформеність: .NET Core дозволяє створювати додатки, що працюють на Windows, Linux та macOS, що робить програму доступною для ширшого кола користувачів.
- Висока продуктивність: .NET Core забезпечує високу продуктивність та швидкість виконання коду.
- Активна підтримка та оновлення: .NET Core активно підтримується Microsoft і спільнотою, що забезпечує постійні оновлення та вдосконалення.
- Можливості масштабування: .NET Core підтримує створення

масштабованих додатків, що дозволяє легко розширювати програму у майбутньому.

Технологія створення інтерфейсу користувача: Windows Presentation Foundation (WPF)

Вибір: Для розробки графічного інтерфейсу користувача вибрано технологію Windows Presentation Foundation (WPF).

Обґрунтування:

- Багаті можливості для створення інтерфейсу: WPF підтримує створення сучасних, багатофункціональних інтерфейсів з використанням XAML, що дозволяє створювати динамічні та привабливі UI.
- Гнучкість та налаштування: WPF надає широкі можливості для кастомізації інтерфейсу, що дозволяє створювати унікальні дизайни.
- Підтримка прив'язки даних (data binding): WPF підтримує прив'язку даних, що спрощує роботу з даними в додатку та робить код чистішим і більш організованим.
- Інтеграція з .NET: WPF інтегрується з .NET, що дозволяє використовувати всі можливості цієї платформи для створення потужних додатків.

Система керування версіями: Git

Вибір: Для керування версіями коду обрано систему Git.

Обґрунтування:

- Популярність та підтримка спільнотою: Git є однією з найпопулярніших систем контролю версій, що забезпечує велику кількість ресурсів та інструментів для роботи.
- Можливість роботи в команді: Git дозволяє ефективно працювати над проектом у команді, забезпечуючи можливість паралельної роботи та злиття змін.
- Безпека та надійність: Git забезпечує надійне зберігання та відстеження історії змін, що дозволяє зберігати всі версії коду і легко повертатися до попередніх станів.

Вибір технологій для розробки програми "Клавіатурний тренажер" ґрунтується на вимогах до сучасного програмного забезпечення, таких як кросплатформеність, висока продуктивність, зручність розробки та підтримка сучасних стандартів. Використання мови C#, платформи .NET Core, технології WPF та системи керування версіями Git забезпечить створення надійного, ефективного та зручного для користувачів програмного продукту.

Висновки до розділу 1

У данному розділі було розглянуто вибір та обґрунтування технологій розробки програмного продукту "Клавіатурний тренажер". Під час аналізу були враховані такі ключові критерії як зручність розробки, можливість підтримки та розширення програми, кросплатформенність та зручність користування. Основними вибраними технологіями були:

Обрано мову програмування C# через її зручність розробки, підтримку об'єктно-орієнтованого програмування, стабільність та продуктивність.

Вибрано платформу .NET Core для кросплатформенності, високої продуктивності, активної підтримки та можливостей масштабування.

Обрано технологію створення інтерфейсу користувача WPF, через багаті можливості для створення сучасного та привабливого інтерфейсу, гнучкість та простоту налаштування, а також підтримку прив'язки даних.

Ці вибори забезпечують ефективну розробку програмного продукту, його кросплатформенність, зручність користування та можливості для майбутнього розширення. Такий підхід до вибору технологій дозволяє створити надійний та функціональний клавіатурний тренажер, що задовольняє потреби користувачів у навчанні швидкому та точному наборі тексту.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

При розробці програми клавіатурного тренажера важливим аспектом є вибір алгоритмів для оцінки та аналізу продуктивності користувача. Основні алгоритми, що використовуються, здійснюють оцінку швидкості набору тексту. Для виміру швидкості набору можуть використовуватися різні методи, такі як кількість натискань клавіш на одиницю часу, середня швидкість набору слова, кількість правильно набраних символів тощо.

Також як алгоритм використовується визначення точності набору. Цей алгоритм полягає в тому що для оцінки точності можуть використовуватися алгоритми, які порівнюють набраний текст з оригінальним текстом та визначають кількість помилок.

Аналіз ефективності тренування також належить до алгоритмів які використовуються при розробці програми. Суть цього алгоритму полягає в тому що деякі алгоритми можуть враховувати час тренування, кількість виконаних вправ та прогрес користувача для надання докладної статистики про його покращення.

Ефективність вибраних алгоритмів полягає у їх здатності точно та об'єктивно виміряти навички користувача, враховуючи різні аспекти набору тексту. Важливо також забезпечити оптимальну продуктивність програми, щоб вона працювала швидко та ефективно навіть при великій кількості даних.

2.2. Структура програми

Ця програма реалізує клавіатурний тренажер і має таку структуру:

Початкові дані і змінні:

- tempTimer - зберігає час початку тренування.
- failCount - лічильник помилок користувача.
- CharCollection і CharLowerCollection - колекції символів для генерації вправ.
- flagCapslock - вказує на вмикнення CapsLock.
- flagBackspase - вказує на натискання клавіші Backspace.
- isStop - вказує на стан тренування (використовується, але не ініціалізується в коді).
- randChar - об'єкт класу Random для генерації випадкових чисел.
- timer - об'єкт DispatcherTimer для вимірювання часу.

Методи для зміни регістру:

- CapitalChar() і LoverChar() - змінюють регістр символів на кнопках.
- CapitalSymbol() і LoverSymbol() - змінюють регістр символів на кнопках зі спеціальними символами.

Обробники подій клавіатури:

- MainWindow_KeyDown() і MainWindow_PreviewKeyUp() - відповідають за зміну регістру символів на кнопках при натисканні клавіш.

Методи для початку та завершення тренування:

- StartButton_Click() і StopButton_Click() - відповідають за початок і завершення тренування.

Генерація вправ:

- GenerateString() - генерує рядок для тренування.

Відстеження продуктивності:

- Timer_Tick() - викликається з інтервалом для вимірювання швидкості набору тексту.
- Speed() - розраховує швидкість набору тексту.

Інші методи:

- StrUser_TextChanged() - викликається при зміні тексту користувачем і відстежує правильність введення.
- DifficultySlider_ValueChanged(), AddCapitalBox_Checked() і AddCapitalBox_Unchecked() - відповідають за налаштування параметрів тренування при зміні користувачем налаштувань.

Завантаження форми:

- Form_Loaded() - викликається при завантаженні вікна і встановлює тестовий час.

Ця структура дозволяє зручно організувати функціонал програми та підтримувати її розширення та підтримку.

Структура проекту показана на рисунку 2.2.1.

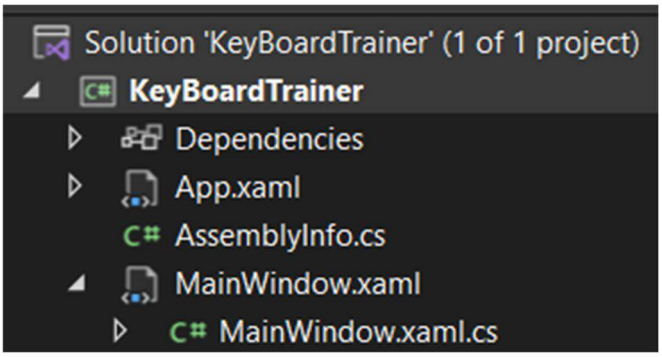

Рисунок 2.2.1. – Структура проекту.

2.3. Програмна реалізація

Програмна реалізація клавіатурного тренажера виконана на мові програмування C# з використанням технології WPF (Windows Presentation Foundation).

Інтерфейс користувача створений за допомогою XAML (eXtensible Application Markup Language), що дозволяє відокремити дизайн від логіки програми. Елементи інтерфейсу, такі як кнопки та тексти, розміщені на головному вікні програми.

Для відслідковування натискання клавіш використовується обробка подій клавіатури. При натисканні клавіш відбувається зміна кольору та значення символів на відповідних кнопках.

При натисканні кнопки "Start" генерується випадковий рядок символів, який користувач повинен буде набирати на клавіатурі.

Програма відстежує кількість помилок користувача, час, витрачений на тренування, і розраховує швидкість набору тексту.

Кнопки "Start" та "Stop" керують початком та завершенням тренування. Параметри тренування, такі як рівень складності та використання великих літер, можна налаштовувати за допомогою слайдера та флажка.

Програма використовує різні класи та бібліотеки для реалізації функціоналу. Наприклад, використання класу Random для генерації випадкових чисел та класу DispatcherTimer для вимірювання часу.

Загалом, програмна реалізація клавіатурного тренажера надає зручний і ефективний спосіб тренування навичок набору тексту на клавіатурі.

2.4. Опис класів

У програмній реалізації клавіатурного тренажера використовуються наступні класи:

MainWindow: Цей клас відповідає за головне вікно програми. Він містить логіку відслідковування подій клавіатури, генерацію вправ, відстеження результатів тренування та управління грою.

TestTime: Цей клас використовується для вимірювання часу тренування. Його методи допомагають визначити, скільки часу користувач витратив на виконання вправи.

Random: Цей клас використовується для генерації випадкових чисел та символів. Він дозволяє програмі створювати випадкові послідовності для вправ тренування.

DispatcherTimer: Цей клас надає можливість встановлювати таймери для виконання певних дій через певні проміжки часу. Використовується для вимірювання швидкості набору тексту та оновлення інтерфейсу програми.

UIElement: Цей клас представляє базовий клас для всіх елементів інтерфейсу користувача в WPF. Використовується для доступу до елементів інтерфейсу та їх взаємодії з користувачем.

Ці класи спільно працюють для забезпечення функціональності програми клавіатурного тренажера, включаючи генерацію вправ, відслідковування подій клавіатури, вимірювання часу та відстеження результатів тренування.

2.5. Тестування програмного продукту

Для перевірки коректності розробки програми «Клавіатурний тренажер» було проведено наступні тести:

Тестування генерації вправ:перевірка того, що вправи генеруються з правильною довжиною; перевірка того, що вправи містять тільки символи, які можуть бути набрані на клавіатурі.

Тестування відповіді на клавіші: перевірка того, що програма відповідає на клавіші коректно, змінюючи відповідність кнопок на інтерфейсі; перевірка того, що програма правильно реагує на введені символи, включаючи відстеження помилок.

Тестування вимірювання часу: перевірка того, що програма правильно вимірює час тренування; перевірка того, що швидкість набору тексту обчислюється правильно.

Тестування відображення результатів: перевірка того, що програма відображає результати тренування коректно, включаючи кількість символів, кількість помилок та швидкість набору тексту.

Тестування функціональності кнопок "Старт" і "Стоп": перевірка того, що кнопка "Старт" активує тренування та вимірює час правильно; перевірка того, що кнопка "Стоп" завершує тренування та відображає результати коректно.

Тестування встановлення рівня складності: перевірка того, що програма правильно встановлює рівень складності відповідно до налаштувань користувача; перевірка того, що програма відображає вправи з відповідною складністю.

Тестування вибору режиму набору тексту: перевірка того, що програма правильно вибирає режим набору тексту (з великими чи малими буквами) відповідно до налаштувань користувача.

Ці тести допомогли переконатися, що програма клавіатурного тренажера працює коректно та надає користувачеві очікувану функціональність.

Висновки до розділу 2

У данному розділі дипломної роботи було розглянуто процес розробки програмного забезпечення клавіатурного тренажера. В ході дослідження було проведено аналіз існуючих аналогів систем клавіатурного тренажера, розглянуто їхні переваги та недоліки. На основі отриманих результатів було сформульовано вимоги до розроблюваної системи.

Для реалізації програми було обрано мову програмування C# та фреймворк .NET Framework, що надає широкі можливості для створення графічних інтерфейсів користувача. Також було вибрано платформу WPF (Windows Presentation Foundation) для розробки графічного інтерфейсу, яка забезпечує зручне та ефективне створення різноманітних користувацьких інтерфейсів.

Структура програми була організована за допомогою різних класів, кожен з яких відповідає за певний функціонал системи. Наприклад, класи для генерації вправ, обробки введеного тексту, вимірювання часу та відображення результатів.

Реалізація програми була випробувана за допомогою набору тестів, які покривають основний функціонал системи та перевіряють її коректну роботу в різних ситуаціях.

Отже, на основі проведеного аналізу та розробки програмного забезпечення можна зробити висновок, що розроблена система клавіатурного тренажера відповідає вимогам, виконує потрібний функціонал та може бути використана для підвищення швидкості та точності набору тексту користувачами.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМНИМ ПРОДУКТОМ

3.1. Мінімальні вимоги до системи

Мінімальні вимоги до системи для коректної роботи клавіатурного тренажера включають:

Операційна система: Windows 7 або новіша.

Процесор: Достатньо потужний для виконання операцій з обробки тексту та графічного інтерфейсу.

Пам'ять: Мінімум 2 ГБ оперативної пам'яті для плавної роботи програми.

Відеокарта: Сумісна з DirectX 10 або новіша для відображення графічного інтерфейсу.

Монітор: Мінімально рекомендована роздільна здатність екрану 1280x800 пікселів.

Вхідні пристрої: Клавіатура для введення тексту та миша для взаємодії з графічним інтерфейсом.

Програмне забезпечення: Microsoft .NET Framework версії 4.5 або новішої для запуску програми.

Програмне забезпечення: Microsoft .NET Framework версії 4.5 або пізнішої, оскільки програма написана на мові програмування C# з використанням технологій WPF (Windows Presentation Foundation).

Місце на диску: Достатньо місця на жорсткому диску для встановлення програми та збереження користувацьких даних. Для більшості програм це може бути дуже невелика кількість місця.

Ці вимоги дозволять користувачам з різними конфігураціями комп'ютерів використовувати клавіатурний тренажер без проблем. Вимоги є мінімальними і можуть варіюватися в залежності від конкретних потреб користувача та функціональних можливостей програми.

3.2. Інтерфейс користувача

Для того щоб відкрити програму користувач має запустити застосунок KeyBoardTrainer. Після чого перед користувачем відкриється вікно програми «Клавіатурний тренажер» що показанео на рисунку 3.2.1.

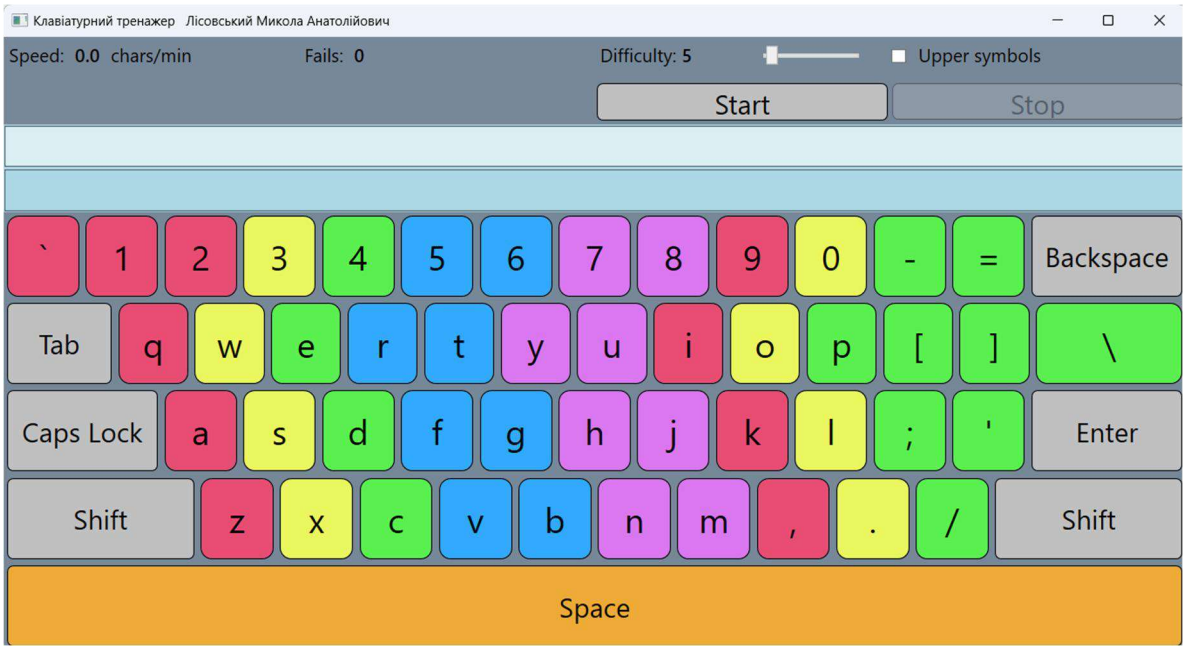

Рис. 3.2.1 Інтерфейс програми

За допомогою повзунка користувач може змінити рівень складності тексту котрий він буде вводити. Це можна зробити за допомогою повзунка як показано на рисунку 3.2.2.

Рис. 3.2.2. Рівень складності тексту.

Користувач може змінити регістр літер котрі будуть вводитись як показано на рисунку 3.2.3.

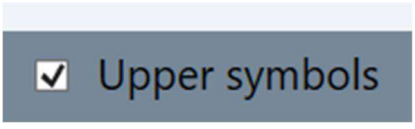

Рис. 3.2.3. Зміна регістру

Швидкість введення тексту за хвилину та кількість помилок які було здійснено при введені тексту. Вигляд показано на рисунку 3.2.4.

Рис. 3.2.4. Швидкість та кількість помилок

Для початку роботи на клавіатурному тренажері для введення тексту користувач має натиснути клавішу Start і для закінчення роботи на тренажері натиснути клавішу Stop відповідно.

Інтерфейс користувача клавіатурного тренажера реалізований з використанням технології Windows Presentation Foundation (WPF), що дозволяє створювати ефективні та привабливі графічні інтерфейси для програм. Ось деякі особливості роботи з інтерфейсом користувача:

Інтерфейс клавіатурного тренажера розроблений з урахуванням зручності та інтуїтивності користування. Елементи керування розташовані логічно та легко розпізнавані, що дозволяє користувачам швидко засвоювати функціонал програми. На інтерфейсі відображаються символи клавіш, які користувач повинен натискати, а також відображається стан різних клавіш, таких як Caps Lock і Shift. Програма відстежує помилки користувача під час набору тексту та відображає їх кількість. Також відстежується час, який користувач витрачає на виконання завдання. Інтерфейс має можливість регулювання складності завдань за допомогою слайдера, що дозволяє змінювати кількість символів для набору. Користувач може вибрати режим, в якому всі символи будуть відображатися великими літерами або ж зі зміною регістру в залежності від стану клавіш Caps Lock і Shift.

Всі елементи інтерфейсу є інтерактивними, що дозволяє користувачам взаємодіяти з програмою за допомогою кліків мишею або натискання клавіш на клавіатурі.

Ці особливості дозволяють забезпечити зручну та ефективну роботу з інтерфейсом користувача клавіатурного тренажера.

Висновки до розділу 3

У цьому розділі дипломної роботи було досліджено та описано програмний продукт - клавіатурний тренажер. Були розглянуті основні вимоги до системи, програмні вимоги, а також розглянуті особливості роботи з інтерфейсом користувача. Детально розглянута програмна реалізація, в якій описано структуру програми, класи, алгоритми та їх ефективність.

Проведено порівняння з аналогічними системами, охарактеризовано вибір та обґрунтування технологій розробки. Також було описано необхідні тести для перевірки коректності програми.

Зазначено, що основні функціональні можливості програми включають в себе генерацію випадкових рядків символів для набору користувачем, відстеження часу та кількості помилок під час набору, а також можливість налаштування складності завдань.

У результаті дослідження було виявлено, що програмний продукт має високий потенціал для використання у навчальних цілях та підвищення швидкості набору тексту користувачами.

ВИСНОВКИ

У дипломній роботі було розроблено програмний продукт - клавіатурний тренажер, призначений для підвищення швидкості набору тексту користувачами. Досліджено та описано всі аспекти розробки програми, включаючи постановку задач, вибір технологій, аналіз аналогічних систем, програмну реалізацію, тести на коректність роботи, а також вимоги до системи.

В результаті проведеного дослідження вдалося успішно розробити та реалізувати клавіатурний тренажер, який відповідає поставленим вимогам та завданням. Програмний продукт має потенціал використання у навчальних цілях для покращення навичок набору тексту та підвищення продуктивності користувачів.

Крім того, в процесі розробки було набуто досвіду у роботі з різноманітними технологіями, виявлено ключові аспекти та вимоги до програмних систем такого типу. Така робота сприяє подальшому професійному зростанню та розвитку у сфері програмної інженерії.

Отже, дипломна робота успішно виконала поставлені цілі та завдання, розширила знання та навички у галузі програмування, а розроблений програмний продукт може бути корисним у навчальних та практичних цілях.

Перелік літературних джерел

1. "Learning to Type Faster and More Accurately: Benefits of Distributed Practice to Lower Error Rates While Typing" by Zachary A. Klase and C. Shawn Green. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3921474/ (accessed on: 2024-06-01).
2. "Keyboarding and Typing Curriculum" by Typing.com. URL: https://www.typing.com/typing-lessons (accessed on: 2024-06-01).
3. "The Importance of Touch Typing" by Keybr.com. URL: https://www.keybr.com/ (accessed on: 2024-06-01).
4. "Effect of keyboard and typing skill on performance of touch screen and non-touch screen smartphone for novice and expert users" by Jeet B. Joshi and Khyati Kakkad. URL: https://www.researchgate.net/publication/287413931_Effect_of_keyboard_and_typing_skill_on_performance_of_touch_screen_and_non-touch_screen_smartphone_for_novice_and_expert_users (accessed on: 2024-06-01).
5. "TypingClub: Learn Touch Typing for Free!" by TypingClub. URL: https://www.typingclub.com/ (accessed on: 2024-06-01).
6. "The Typing Cat: Learn Touch Typing for Free!" by The Typing Cat. URL: https://thetypingcat.com/ (accessed on: 2024-06-01).
7. "Keybr: Improve Your Typing Speed!" by Keybr. URL: https://www.keybr.com/ (accessed on: 2024-06-01).
8. "Rapid Typing: Free Typing Tutor!" by Rapid Typing. URL: https://rapidtyping.com/ (accessed on: 2024-06-01).
9. "Mastering Typing Skills: The Importance of Touch Typing" by TypingTest.com. URL: https://www.typingtest.com/ (accessed on: 2024-06-01).
10. "Touch Typing Lessons and Games" by Sense-Lang.org. URL: https://www.sense-lang.org/typing/ (accessed on: 2024-06-01).
11. "Improve Your Typing Speed with KeyHero" by KeyHero. URL: https://www.keyhero.com/ (accessed on: 2024-06-01).

12. "Speed Typing Online" by SpeedTypingOnline. URL: https://www.speedtypingonline.com/ (accessed on: 2024-06-01).
13. "TypeRacer: The Global Typing Competition!" by TypeRacer. URL: https://play.typeracer.com/ (accessed on: 2024-06-01).
14. "Keybricks: Learn Touch Typing Online!" by Keybricks. URL: https://keybricks.org/ (accessed on: 2024-06-01).
15. "TypingAcademy: Learn Touch Typing!" by TypingAcademy. URL: https://www.typing.academy/ (accessed on: 2024-06-01).
16. "PowerTyping: Free Online Typing Tutor!" by PowerTyping. URL: https://www.powertyping.com/ (accessed on: 2024-06-01).
17. "TypingGames.zone: Play Free Typing Games!" by TypingGames.zone. URL: https://www.typinggames.zone/ (accessed on: 2024-06-01).
18. "TypingBee: Online Typing Tutor and Typing Test!" by TypingBee. URL: https://www.typingbee.com/ (accessed on: 2024-06-01).
19. "Touch Typing Study" by Touch Typing Study. URL: https://www.touchtypingstudy.com/ (accessed on: 2024-06-01).
20. "Keybr.com: Touch Typing Lessons" by Keybr.com. URL: https://www.keybr.com/ (accessed on: 2024-06-01).

ДОДАТКИ

Додаток А

Програмний код модулів

```
using System;
using System.Data;
using System.Reflection;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Threading;
using workfile;
namespace KeyBoardTrainer {
    public partial class MainWindow : Window     {
        DateTime tempTimer;
        int failCount = 0;
        string                                CharCollection                                =
"qwertyuiopasdfghjklzxcvbnm~1234567890!@#$%^&*()_+{}|:\"<>?[],./\\`-
=;'QWERTYUIOPASDFGHJKLZXCVBNM";
        string              CharLowerCollection          =          "`1234567890-
=qwertyuiop[]asdfghjkl;'zxcvbnm,./\\";
        bool flagCapslock = false;
        bool flagBackspase = true;
        bool isStop = true;
        Random randChar = new Random();
        DispatcherTimer? timer = null;
        public MainWindow()        {
            InitializeComponent();
            timer = new DispatcherTimer();
            timer.Tick += Timer_Tick;
```

```
        timer.Interval = new TimeSpan(0, 0, 0, 0, 200);         }
    private void CapitalChar()         {
        this.Q.Content = "Q";
        this.W.Content = "W";
        this.E.Content = "E";
        this.R.Content = "R";
        this.T.Content = "T";
        this.Y.Content = "Y";
        this.U.Content = "U";
        this.I.Content = "I";
        this.O.Content = "O";
        this.P.Content = "P";
        this.A.Content = "A";
        this.S.Content = "S";
        this.D.Content = "D";
        this.F.Content = "F";
        this.G.Content = "G";
        this.H.Content = "H";
        this.J.Content = "J";
        this.K.Content = "K";
        this.L.Content = "L";
        this.Z.Content = "Z";
        this.X.Content = "X";
        this.C.Content = "C";
        this.V.Content = "V";
        this.B.Content = "B";
        this.N.Content = "N";
        this.M.Content = "M";         }
    private void LoverChar()         {
        this.Q.Content = "q";
```

```
            this.W.Content = "w";
            this.E.Content = "e";
            this.R.Content = "r";
            this.T.Content = "t";
            this.Y.Content = "y";
            this.U.Content = "u";
            this.I.Content = "i";
            this.O.Content = "o";
            this.P.Content = "p";
            this.A.Content = "a";
            this.S.Content = "s";
            this.D.Content = "d";
            this.F.Content = "f";
            this.G.Content = "g";
            this.H.Content = "h";
            this.J.Content = "j";
            this.K.Content = "k";
            this.L.Content = "l";
            this.Z.Content = "z";
            this.X.Content = "x";
            this.C.Content = "c";
            this.V.Content = "v";
            this.B.Content = "b";
            this.N.Content = "n";
            this.M.Content = "m";         }
        private void CapitalSymbol()        {
            this.Oem3.Content = "~";
            this.D1.Content = "!";
            this.D2.Content = "@";
            this.D3.Content = "#";
```

```
            this.D4.Content = "$";
            this.D5.Content = "%";
            this.D6.Content = "^";
            this.D7.Content = "&";
            this.D8.Content = "*";
            this.D9.Content = "(";
            this.D0.Content = ")";
            this.OemMinus.Content = "_";
            this.OemPlus.Content = "+";
            this.OemOpenBrackets.Content = "{";
            this.Oem6.Content = "}";
            this.Oem5.Content = "|";
            this.Oem1.Content = ":";
            this.OemQuotes.Content = "\"";
            this.OemComma.Content = "<";
            this.OemPeriod.Content = ">";
            this.OemQuestion.Content = "?";        }
        private void LoverSymbol()        {
            this.Oem3.Content = "`";
            this.D1.Content = "1";
            this.D2.Content = "2";
            this.D3.Content = "3";
            this.D4.Content = "4";
            this.D5.Content = "5";
            this.D6.Content = "6";
            this.D7.Content = "7";
            this.D8.Content = "8";
            this.D9.Content = "9";
            this.D0.Content = "0";
            this.OemMinus.Content = "-";
```

```
            this.OemPlus.Content = "=";
            this.OemOpenBrackets.Content = "[";
            this.Oem6.Content = "]";
            this.Oem5.Content = "\\";
            this.Oem1.Content = ";";
            this.OemQuotes.Content = "'";
            this.OemComma.Content = ",";
            this.OemPeriod.Content = ".";
            this.OemQuestion.Content = "/";        }
        private void MainWindow_KeyDown(object sender, KeyEventArgs e)        {
            foreach (UIElement grid in ((Grid)Content).Children)            {
                if (grid is Grid)                {
                    foreach (var item in ((Grid)grid).Children)
        {
                        if (item is Button)
            {
                            if (((Button)item).Name == e.Key.ToString())
                        {
                                ((Button)item).Opacity = 0.3;
                                if  (e.Key.ToString()  ==  "LeftShift"  ||  e.Key.ToString()  ==
"RightShift")
            {
                                    CapitalSymbol();
                                    if (!flagCapslock)
                            {
                                        CapitalChar();
                            }
                                    Else
        {
                                        LoverChar();
```

```
                }
            }
                    if (e.Key.ToString() == "Capital")
                {
                    if (!flagCapslock)
            {
                        CapitalChar();
                        flagCapslock = true;                    }
                    else                        {
                        LoverChar();
                        flagCapslock = false;                    }                }
                    else if (e.Key.ToString() == "Back")                    {
                        flagBackspase = false;                    }
                    else                    {
                        flagBackspase = true;       }       }
        private void MainWindow_PreviewKeyUp(object sender, KeyEventArgs e)   {
            foreach (UIElement grid in ((Grid)Content).Children)          {
              if (grid is Grid)             {
                foreach (var item in ((Grid)grid).Children)             {
                  if (item is Button)                   {
                    if (((Button)item).Name == e.Key.ToString())                    {
                      ((Button)item).Opacity = 1;
                      if  (e.Key.ToString()  ==  "LeftShift"  ||  e.Key.ToString()  ==
"RightShift")                    {
                        LoverSymbol();
                        if (!flagCapslock)                    {
                          LoverChar();                  }
                        Else                    {
                          CapitalChar();                    }
        private void StartButton_Click(object sender, RoutedEventArgs e)
```

```
{           StartButton.IsEnabled = false;
    StartButton.Opacity = 0.3;
    StopButton.IsEnabled = true;
    StopButton.Opacity = 1;
    DifficultySlider.IsEnabled = false;
    DifficultySlider.Opacity = 0.3;
    AddCapitalBox.IsEnabled = false;
    AddCapitalBox.Opacity = 0.3;
    strUser.IsReadOnly = false;
    strUser.IsEnabled = true;
    GenerateString();          timer?.Start();          tempTimer = DateTime.Now;
    strUser.Focus();       }
  private void StopButton_Click(object sender, RoutedEventArgs e)       {
    timer?.Stop();
    MessageBox.Show($"Task completed!\n" +
                $"Symbol count: {strProgram.Text.Length}.\n" +
                $"Fails count: {FailsCount.Content}.\n" +
                $"Time: {(DateTime.Now - tempTimer).Seconds:0.0}s.\n" +
                $"Symbols per minute: {SpeedChar.Content}.\n",
                "Results",                                    MessageBoxButton.OK,
MessageBoxImage.Information);          StartButton.IsEnabled = true;
    StartButton.Opacity = 1;
    StopButton.IsEnabled = false;
    StopButton.Opacity = 0.3;          DifficultySlider.IsEnabled = true;
    DifficultySlider.Opacity = 1;
    AddCapitalBox.IsEnabled = true;
    AddCapitalBox.Opacity = 1;
    strUser.IsReadOnly = true;
    strUser.IsEnabled = false;
    strUser.Text = "";
```

```
            strProgram.Text = "";
            FailsCount.Content = 0;
            SpeedChar.Content = "" + 0;
            failCount = 0;
        }
        private void Timer_Tick(object? sender, EventArgs e)
        {
            Speed();        }
        private void GenerateString()        {
            string TempRandomStr = "";
            if (AddCapitalBox.IsChecked == true)            {
                for (int i = 0; i < Convert.ToInt32(DifficultyLevel.Content); i++)            {
                    TempRandomStr            +=            CharCollection[randChar.Next(0,
CharCollection.Length)];
                }
        }
        else
        {
                for (int i = 0; i < Convert.ToInt32(DifficultyLevel.Content); i++)            {
                    TempRandomStr        +=        CharLowerCollection[randChar.Next(0,
CharLowerCollection.Length)];
        }           }
            TempRandomStr += " ";
            for (int i = 0; i < 75; i++)
        {
                strProgram.Text            +=            TempRandomStr[randChar.Next(0,
TempRandomStr.Length)];
        }       }
        private void StrUser_TextChanged(object sender, TextChangedEventArgs e)  {
            string str = strProgram.Text.Substring(0, strUser.Text.Length);
```

```
            if (strUser.Text.Equals(str))           {
                if (flagBackspase)
        {
        Speed();
          }
                strUser.Foreground = new SolidColorBrush(Colors.Green);
       }
            else
       {
                if (flagBackspase)
        {
           failCount++;
          }
                strUser.Foreground = new SolidColorBrush(Colors.DarkRed);
                FailsCount.Content = failCount;
        }
            if (strUser.Text.Length >= strProgram.Text.Length || strUser.Text ==
strProgram.Text)
      {
                strUser.IsReadOnly = true;
                Speed();
                StopButton_Click(null, null);
       }       }
        private void DifficultySlider_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)       {
            Slider s = (Slider)sender;
            DifficultyLevel.Content = (int)s.Value;       }
        private void AddCapitalBox_Checked(object sender, RoutedEventArgs e)      {
            DifficultySlider.Maximum = 94;       }
        private void AddCapitalBox_Unchecked(object sender, RoutedEventArgs e)
```

```
        {           DifficultySlider.Maximum = 47;
    }
      void Speed()
   {
         SpeedChar.Content = $"{((strUser.Text.Length * 60.0) / (DateTime.Now -
tempTimer).Seconds):N3}";
    }
      private void Form_Loaded(object sender, RoutedEventArgs e)
    {
              }}
```