Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
«___»_____202_ р.

**КВАЛІФІКАЦІЙНА РОБОТА**
**на тему**

# АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ ТРЕНАЖЕРА «РЕГУЛЯРНІ ВИРАЗИ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАММУВАНЯ»

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

**Виконавець роботи** _Сироткін Олексій Олегович_
_____«___»_____2023р.
(підпис)

**Науковий керівник** _к.ф.-м.н., доц. Черненко Оксана Олексіївна_
_____ «___»_____2023р.
(підпис)

**Рецензент** _____

**ПОЛТАВА 2023р.**

**ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ**
**«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»**

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри** _О.В. Ольховська_
«__ » _вересня_202_ р.

**Завдання та календарний графік**
**виконання кваліфікаційної роботи**

**Студент спеціальності 122 «Комп'ютерні науки»**
**Прізвище, ім'я, по батькові Сироткін Олексій Олегович**

1. Тема **«Алгоритмізація та програмування тренажера «Регулярні вирази» дистанційного навчального курсу «Теорія програмування» »**
затверджена наказом ректора № 144-Н від «__» _____ 202__ р.
Термін подання студентом бакалаврської роботи «___» _____202__ р.

2. Вихідні дані до бакалаврської роботи: методичні матеріали за темою роботи;
матеріали дистанційного навчального курсу «Теорія програмування» та стандарти.

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

ВСТУП

1. ПОСТАНОВКА ЗАДАЧІ

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Огляд схожих за тематикою тренажерів

2.2 Актуальність розробки тренажеру

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 РЕГУЛЯРНІ ВИРАЗИ

3.2 Алгоритм роботи тренежера

3.3 Блок-схеми програми тренажера

4. ПРАКТИЧНА ЧАСТИНА

4.1 Опис розробки тренажера

4.2 Інструкція до роботи з тренажером

ВИСНОВОК

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А

4. Перелік графічного матеріалу: 3-4 аркуші блок-схем, інші необхідні ілюстрації.

5. Консультанти розділів бакалаврської роботи

| Розділ | Прізвище, ініціали, посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1. Постановка задачі | Черненко О.О. | | |
| 2. Інформаційний огляд | Черненко О.О. | | |
| 3. Теоретична частина | Черненко О.О. | | |
| 4. Практична реалізація | Черненко О.О. | | |

6. Календарний графік виконання бакалаврської роботи

| Зміст роботи | Термін виконання | Фактичне виконання |
|---|---|---|
| 1. Вступ | | |
| 2. Вивчення методичних рекомендацій та стандартів та звіт керівнику | | |
| 3. Постановка задачі | | |
| 4. Інформаційний огляд джерел бібліотек та інтернету | | |
| 5. Теоретична частина | | |
| 6. Практична частина | | |
| 7. Закінчення оформлення | | |
| 8. Доповідь студента на кафедрі | | |
| 9. Доробка (за необхідністю), рецензування | | |

Дата видачі завдання «__» _____ 202_ р.

Студент Сироткін Олексій Олегович

Науковий керівник к.ф.-м.н., доц. Черненко Оксана Олексіївна

***Результати захисту кваліфікаційної роботи***

Кваліфікаційної робота оцінена на

_____

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

Протокол засідання ЕК № ____ від «____» _____ 202_ р.

Секретар ЕК_____ _____

(підпис) (ініціали та прізвище)

# РЕФЕРАТ

**Записка:** 44 стор., в т.ч. основна частина 40 стор., джерел - 7.

Мета роботи: розробка програмного забезпечення тренажеру з теми «Регулярні вирази» для дистанційного навчального курсу «Теорія програмування».

Об’єкт розробки: програма-тренажер на тему «Регулярні вирази».

Предмет розробки: програмне забезпечення тренажеру з теми «Регулярні вирази» для дистанційного навчального курсу «Теорія програмування».

Методи розробки: для програмної реалізації тренажеру використано середовище розробки MS Visual Studio Code за допомогою мови програмування Python 3.10 та використання бібліотеки PyQT5.

Методи дослідження: При розробці тренажера з регулярними виразами для дистанційного навчального курсу "Теорія програмування" можуть бути використані наступні методи дослідження:

- Літературний огляд: Дослідження та аналіз наявних джерел, книг, наукових статей, документації, інтернет-ресурсів, пов'язаних з регулярними виразами, алгоритмізацією та програмуванням. Цей метод дозволяє збирати інформацію про теоретичні аспекти, приклади використання та рекомендації щодо розробки тренажера.
- Тестування та зворотній зв'язок: Випробування тренажера з регулярними виразами серед цільової аудиторії, яка складається з учнів курсу "Теорія програмування". Збір поверненого зворотного зв'язку, аналіз результатів тестування та впровадження виправлень та вдосконалень на основі отриманих даних

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

| Умовні позначення, символи, скорочення, терміни | Пояснення умовного позначення, скорочень, символів |
|---|---|
| Тренажер | Програма-вчитель, призначена для вивчення і закріплення знань з різноманітних тем; створюється для різноманітних операційних систем за бажанням та необхідністю. |
| *ДК* | Дистанційний курс. |
| *ДН* | Дистанційне навчання. |
| *Python* | Мова програмування. |
| *PyQt5* | Бібліотека Python |
| *IDE Visual Studio Code* | Середовище для написання коду. |

# ВСТУП

В даний час зростає значущість вивчення регулярних виразів в контексті програмування та розробки програмного забезпечення. Регулярні вирази є потужним інструментом для роботи з текстовими даними, пошуку та заміни певних шаблонів. Вони знаходять застосування в різних галузях, таких як обробка даних, валідація введення користувача, аналіз лог-файлів, машинне навчання та багато інших.

Однак, навчитися ефективно використовувати регулярні вирази може бути складним завданням для студентів і початківців, особливо тих, хто навчається на дистанційному курсі. Брак особистого контакту з викладачем та відсутність можливості отримати миттєву зворотну зв'язок можуть ускладнювати процес навчання регулярним виразам.

Тому, з метою полегшити процес навчання та розуміння регулярних виразів, пропонується розробити тренажер «Регулярні вирази» для дистанційного навчального курсу «Теорія програмування». Тренажер буде забезпечувати студентам можливість вчитися та практикуватися в роботі з регулярними виразами, надавати зворотній зв'язок та підтримку в процесі навчання.

Мета кваліфікаційної роботи полягає в розробці та реалізації тренажера «Регулярні вирази» для дистанційного навчального курсу «Теорія програмування». Основні завдання роботи включають:

- Аналіз вимог до тренажера «Регулярні вирази» з урахуванням особливостей дистанційного навчання та навчального курсу «Теорія програмування».
- Проектування інтерфейсу користувача тренажера, забезпечення зручності використання та навігації для студентів.
- Розробка алгоритмів перевірки та валідації регулярних виразів, забезпечення коректності та точності результатів.

- Реалізація тренажера з використанням відповідних технологій та програмних засобів.
- Тестування тренажера та оцінка його ефективності та якості роботи.

Результатом кваліфікаційної роботи буде функціональний тренажер «Регулярні вирази», який буде допомагати студентам в освоєнні та практичному застосуванні регулярних виразів у програмуванні. Також, робота сприятиме поліпшенню процесу навчання на дистанційному курсі та підвищенню якості освіти з теми «Теорія програмування».

Обсяг пояснювальної записки: 44 сторінки, з них основна частина – 40 сторінок, додатки – 2 сторінки, інформаційні джерела – 12 назв.

# 1. ПОСТАНОВКА ЗАДАЧІ

В рамках кваліфікаційної роботи поставлено наступні задачі:

1. Аналіз вимог до тренажера «Регулярні вирази»:
   1.1. Визначення функціональних вимог, які повинен задовольняти тренажер. Це можуть бути такі функції, як введення регулярного виразу, валідація його правильності, можливість виконання пошуку або заміни шаблонів у текстових даних тощо.
   1.2. Визначення нефункціональних вимог, таких як зручність використання, швидкодія, надійність, підтримка різних платформ тощо.
2. Проектування інтерфейсу користувача тренажера:
   2.1. Розробка зручного та інтуїтивно зрозумілого інтерфейсу, який дозволить користувачам вводити регулярні вирази, виконувати операції з ними та отримувати результати.
   2.2. Розробка ефективної системи навігації та взаємодії з тренажером, забезпечення зручного доступу до всіх функцій і можливостей.
3. Розробка алгоритмів перевірки та валідації регулярних виразів:
   3.1. Реалізація алгоритмів перевірки синтаксичної правильності введеного регулярного виразу.
   3.2. Розробка алгоритмів перевірки коректності та валідності регулярних виразів залежно від вимог і контексту застосування.
4. Реалізація тренажера «Регулярні вирази»:
   4.1. Вибір відповідних технологій та програмних засобів для реалізації тренажера.
   4.2. Розробка архітектури та структури програмного продукту.
   4.3. Реалізація основних функцій тренажера, включаючи введення регулярних виразів, виконання операцій з текстовими даними та виведення результатів.
5. Тестування тренажера та оцінка його ефективності та якості роботи:

5.1. Проведення різноманітних тестів для перевірки коректності та надійності тренажера.

5.2. Оцінка швидкодії та продуктивності тренажера при обробці регулярних виразів та текстових даних.

5.3. Збір фідбеку від користувачів та врахування їх пропозицій для покращення тренажера.

Метою роботи є розробка тренажера «Регулярні вирази» для дистанційного навчального курсу «Теорія програмування», який допоможе студентам ефективно навчитися та практикуватися в роботі з регулярними виразами.

# 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

## 2.1 Огляд схожих за інформаційним складом тренажерів

На сьогоднішній день існує кілька тренажерів, спрямованих на навчання та практикування роботи з регулярними виразами. Деякі з них можуть бути корисними для огляду та вивчення функцій, які можуть бути реалізовані у тренажері "Регулярні вирази" для дистанційного навчального курсу "Теорія програмування". Нижче наведено огляд деяких схожих тренажерів:

- RegexOne (https://regexone.com/) - це онлайн-тренажер, який надає можливість вивчати регулярні вирази шляхом виконання послідовних завдань. Кожне завдання полягає в написанні регулярного виразу, який відповідає певному шаблону. Тренажер надає наглядні пояснення та приклади для допомоги студентам у розумінні концепцій регулярних виразів.
- RegExr (https://regexr.com/) - це інтерактивний редактор регулярних виразів, який дозволяє користувачам створювати, тестувати та візуалізувати регулярні вирази в режимі реального часу. Він надає можливість вводити текстові дані та застосовувати регулярні вирази до них, відображаючи збіги та заміни.
- Rubular (https://rubular.com/) - це онлайн-тренажер для регулярних виразів, спеціально спрямований на мову програмування Ruby. Він дозволяє користувачам вводити регулярні вирази та тестувати їх на текстових даних. Результати відображаються в реальному часі, що допомагає студентам відстежувати та розуміти відповідність між регулярними виразами та текстом.

При розробці тренажера "Регулярні вирази" для дистанційного навчального курсу "Теорія програмування" можна аналізувати та використовувати переваги та недоліки цих схожих тренажерів а саме:

1. Обмежена кількість завдань - RegexOne пропонує послідовні завдання для виконання, але їх кількість може бути обмеженою. Це може

призвести до того, що студенти швидко переборюють всі завдання і не мають достатньої практики.

2. Складний інтерфейс - RegExr має візуально насичений інтерфейс з багатьма налаштуваннями та панелями. Це може бути побічним чинником, що ускладнює початкове користування та розуміння тренажера для новачків.

3. Відсутність оновлень - Розробка та оновлення Rubular можуть бути обмеженими, що може вплинути на його сумісність з останніми версіями Ruby або бажання користувачів отримати нові функції та поліпшення.

## 2.2 Актуальність розробки тренажеру

Дослідивши обрані для інформаційного огляду тренажери було знайдено як і позитивні сторони так і багато негативних аспектів. Вважається доцільним та актуальним розробка тренажеру, що буде поєднувати в собі позитивні сторони, побачені в інших тренажерах, та уникатиме помилок, помічених в інших тренажерах.

# 3. ТЕОРЕТИЧНА ЧАСТИНА

## 3.1 РЕГУЛЯРНІ ВИРАЗИ

Модуль re з'явився у версії Python 1.5 і був призначений для роботи з регулярними виразами у стилі Perl. У більш старих версіях Python використовувався модуль regex, який втілював регулярні вирази у стилі Emacs. Але з версії Python 2.5 модуль regex був вилучений.

Регулярні вирази (скорочено RE або regex) - це маленька мова програмування, яка вбудована в Python за допомогою модуля re. Ця мова дозволяє описувати множини рядків, які відповідають певному шаблону (регулярному виразу). За допомогою регулярних виразів можна визначати правильні електронні адреси, команди TeX або будь-який інший шаблон. За допомогою цих виразів можна відповідати на питання, чи відповідає певний рядок заданому шаблону, або чи міститься у рядку певна послідовність символів, що відповідає шаблону. Регулярні вирази також можна використовувати для заміни певних підрядків у рядку або для розбиття рядка на складові.

Регулярні вирази компілюються в послідовність байткодів, які потім виконуються підпрограмою, написаною на мові C.

Однак, через те, що регулярні вирази є обмеженою мовою, вони не підходять для всіх завдань обробки рядків. Деякі задачі можуть стати занадто складними для вирішення за допомогою регулярних виразів, і в таких випадках ефективніше написати власний код обробки рядків на Python.

Окрім того, що регулярні вирази є обмеженою мовою, вони також можуть бути важкими для розуміння і підтримки в майбутньому. Коли регулярні вирази стають складними і включають багато спеціальних символів та правил, вони можуть стати заплутаними і складними для відлагодження. Це може призвести до проблем з читабельністю і розумінням коду з регулярними виразами.

Крім того, обробка рядків за допомогою регулярних виразів може бути менш ефективною в порівнянні з використанням вбудованих методів рядків Python. Регулярні вирази можуть вимагати багато ресурсів і часу обчислень для складних шаблонів, особливо при роботі з великими обсягами даних. В таких випадках написання власного коду обробки рядків на Python може бути швидшою і ефективнішою альтернативою.

Отже, регулярні вирази є потужним і корисним інструментом для обробки рядків, проте вони не є універсальним рішенням для всіх завдань. В деяких випадках, особливо при складних шаблонах або великих обсягах даних, може бути доцільніше використовувати інші підходи до обробки рядків у Python.

## 3.2 Алгоритм роботи тренежера

Робота з програмним забезпеченням виконується за наступним алгоритмом:

Етап 1. Запуск програмного забезпечення.

Етап 2. Користувачу надається доступ до початкового меню тренажеру, в якому видно тему та розробника роботи.

Етап 3. На початковому меню тренажера за допомогою кнопки «Розпочати» користувач переходить до практичних завдань.

Етап 4. Користувачу надається практичне завдання у форматі тесту.:

Завдання 1. Яка функція метасимволу \d в регулярних виразах?

a) Відповідає будь-якій цифрі

b) Відповідає будь-якому символу

c) Відповідає початку рядка

d) Відповідає кінцю рядка

Вірний варіант: a)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 5. Користувачу надається практичне завдання у форматі тесту.:

Завдання 2. Який оператор використовується для повторення попереднього символу один або більше разів?

a) ?

b) +

c) *

d) |

Вірний варіант: b)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 6. Користувачу надається практичне завдання у форматі тесту.:

Завдання 3. Як в регулярних виразах позначається початок рядка?

a) ^

b) $

c) *

d) .

Вірний варіант: a)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 7. Користувачу надається практичне завдання у форматі тесту.:

Завдання 4. Яка послідовність символів використовується для вибору одного з декількох символів?

a) ()

b) {}

c) []

d) //

Вірний варіант: c)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 8. Користувачу надається практичне завдання у форматі тесту.:

Завдання 5. Яка функція метасимволу \s в регулярних виразах?

a) Відповідає пробілу

b) Відповідає будь-якій цифрі

c) Відповідає будь-якому символу

d) Відповідає початку рядка

Вірний варіант: a)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 9. Користувачу надається практичне завдання у форматі тесту.:

Завдання 6. Який оператор використовується для вибору між двома альтернативними варіантами?

a) ?

b) +

c) *

d) |

Вірний варіант: d)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 10. Користувачу надається практичне завдання у форматі тесту.:

Завдання 7. Як в регулярних виразах позначається кінець рядка?

a) ^

b) $

c) *

d) .

Вірний варіант: b)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 11. Користувачу надається практичне завдання у форматі тесту.:

Завдання 8. Яка функція метасимволу \w в регулярних виразах?

a) Відповідає пробілу

b) Відповідає будь-якій літері алфавіту

c) Відповідає будь-якій цифрі

d) Відповідає будь-якому символу

Вірний варіант: b)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 12. Користувачу надається практичне завдання у форматі тесту.:

Завдання 9. Який оператор використовується для повторення попереднього символу нуль або один раз?

a) ?

b) +

c) *

d) |

Вірний варіант: a)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 13. Користувачу надається практичне завдання у форматі тесту.:

Завдання 10. Яка функція метасимволу \b в регулярних виразах?

a) Відповідає пробілу

b) Відповідає початку слова

c) Відповідає будь-якій літері алфавіту

d) Відповідає кінцю рядка

Вірний варіант: b)

Після успішного відповідання користувач має можливість натиснути кнопку «Далі», яка перенаправить його до наступного практичного завдання.

Етап 14. Після завершення виконання практичних матеріалів тренажеру, користувач переходить до фінального екрану, де він має можливість завершити або повторити свою роботу за допомогою відповідних кнопок.

## 3.3 Блок-схеми алгоритму тренажера

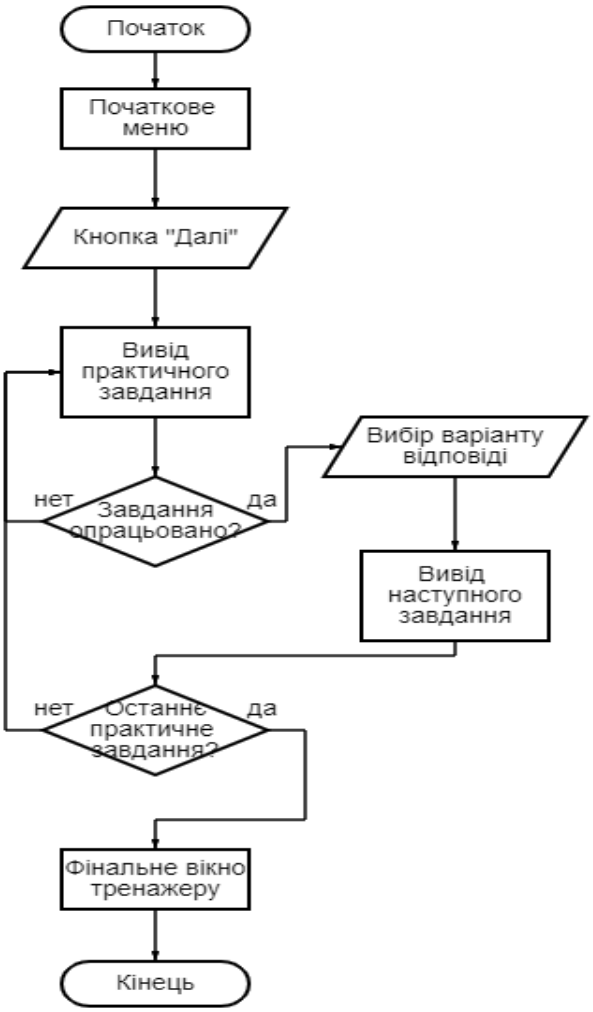



Рисунок 3.1 – Блок-схеми алгоритму роботи тренажера

# 4. ПРАКТИЧНА ЧАСТИНА

## 4.1 Опис розробки тренажера

Вибір Python та бібліотеки PyQt для створення тренажера з регулярними виразами базується на кількох факторах.

По-перше, Python є однією з найпопулярніших та найрозповсюдженіших мов програмування. Вона має простий і зрозумілий синтаксис, що дозволяє легко вивчати мову навіть початківцям. Python також має широкий спектр бібліотек та інструментів, що спрощують розробку програм. Вибір Python забезпечує гнучкість та продуктивність для розробки тренажера з регулярними виразами.

По-друге, бібліотека PyQt є потужним інструментом для розробки графічних інтерфейсів користувача в мові Python. Вона забезпечує багато можливостей для створення інтерактивних програм з вікнами, кнопками, текстовими полями та іншими елементами інтерфейсу. PyQt також має велику спільноту розробників, яка надає підтримку, документацію та приклади коду для полегшення процесу розробки.

Об'єднання Python і PyQt дозволяє створити потужний та зручний тренажер з регулярними виразами. Python надає можливість легко розробляти логіку та обробляти дані, а PyQt допомагає створити ефективний та привабливий графічний інтерфейс для взаємодії з користувачем. Вибір Python та PyQt є оптимальним для реалізації тренажера з регулярними виразами з урахуванням простоти вивчення, гнучкості та розширюваності.

За допомогою додаткової програми Qt Designer ми створюємо сам дизайн вікна, де саме буде знаходитись питання та відповіді на варіанти відповідей (див. Рисунок 4.1):

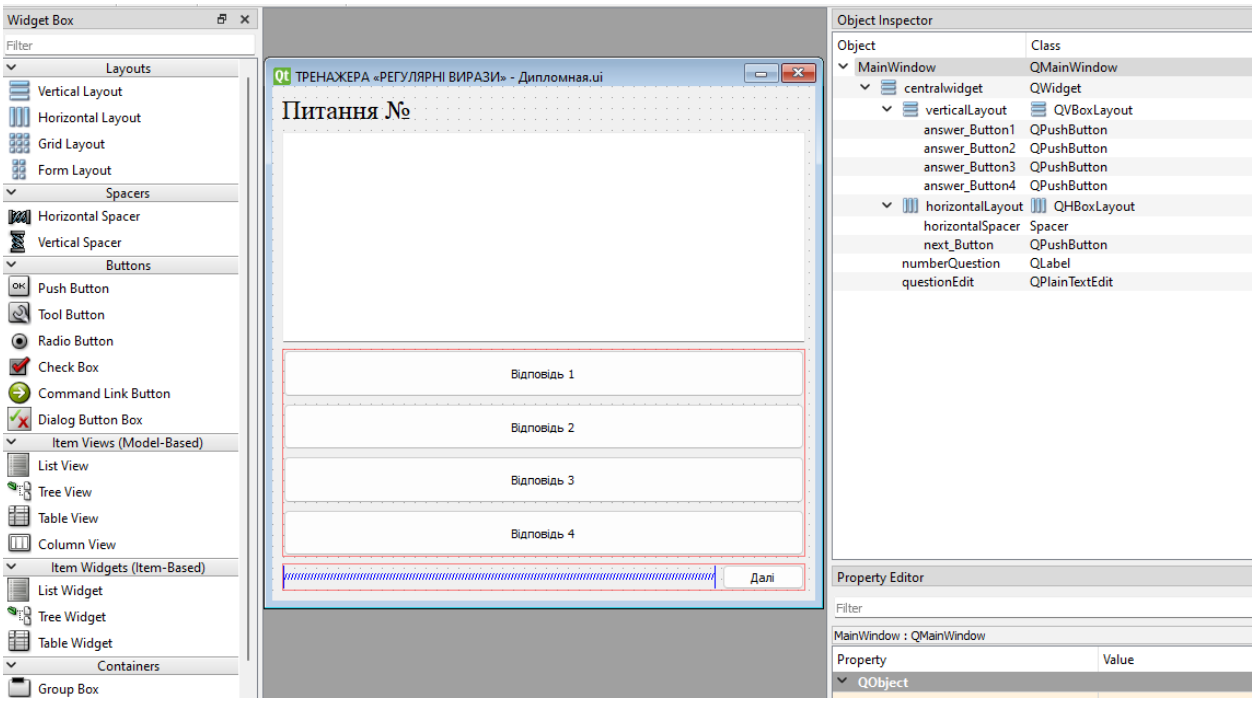
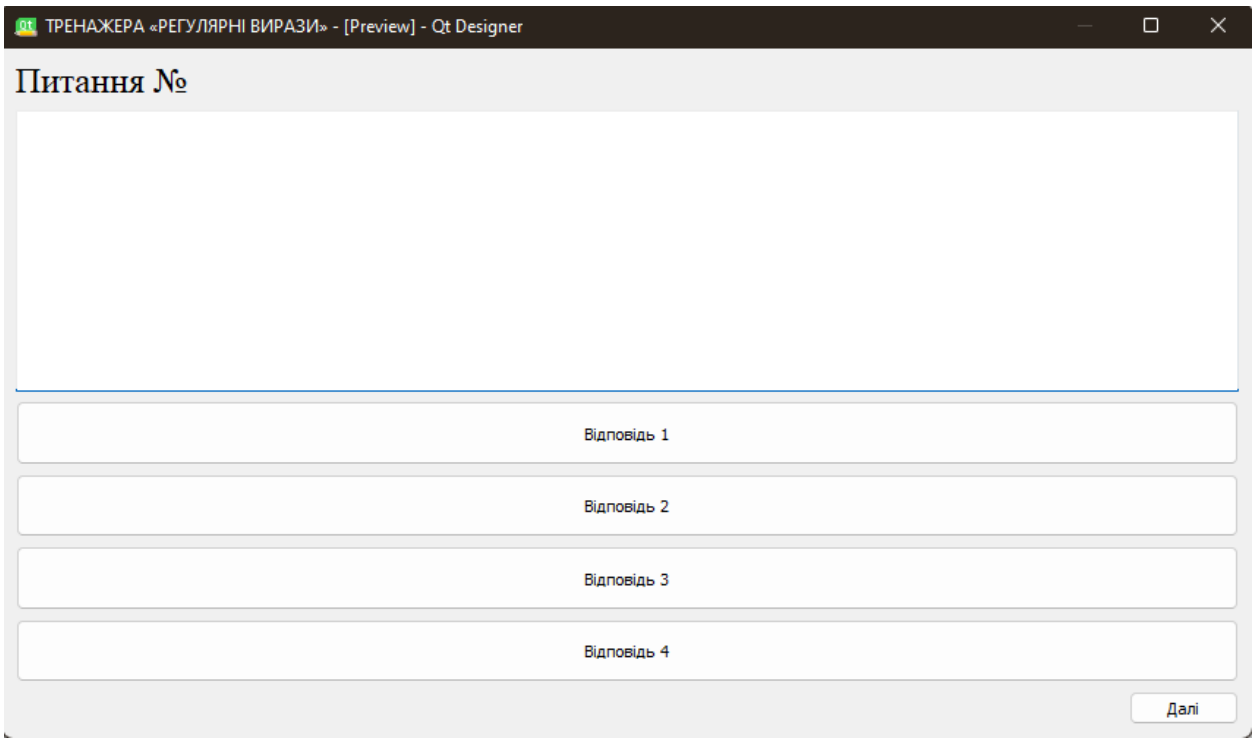



Рисунок 4.1 – Створення вікна де будуть знаходитись питання.

За допомогою елементів керування, додаємо рамку де буде знаходитись текст питання, додаємо елемент де буде виводитись саме питання та додаємо текст з номером питанням, щоб користувач знав на якому він питанні. Розмір програми є адаптивним, тобто користувач маємо



можливість його розтягнути як хоче. (див. Рисунок 4.2):

Рисунок 4.2. Показ адаптивності програми

В таблиці розписані всі елементи керування які було використано.

| **Елемент керування** | **Назва елементу в QT** |
|---|---|
| Кнопка | QPushButton |
| Текст | QLabel |
| Вікно питання | QPlainTextEdit |

Кнопка «Далі» буде відповідати за переключення на наступне питання, якщо на питання було дано правильну відповідь, якщо користувач відповідає неправильно, то програма повідомляє користувачу, що це не вірна відповідь, і очікує на правильну відповідь.

Після закінчення роботи з тренажером користувач отримує доступ до фінального вікна у якому може повторити роботу за допомогою відповідної кнопки. Кнопка «Повторити» працює ідентично до кнопок «Далі», але вона «вимикає» останній слайд в тренажері та «вмикає» початкове меню, що дозволяє повторити роботу з тренажером.

## 4.2 Інструкція до роботи з тренажером

Після запуску на початковому меню тренажера за допомогою кнопки «Розпочати» користувач переходить до практичних завдань.

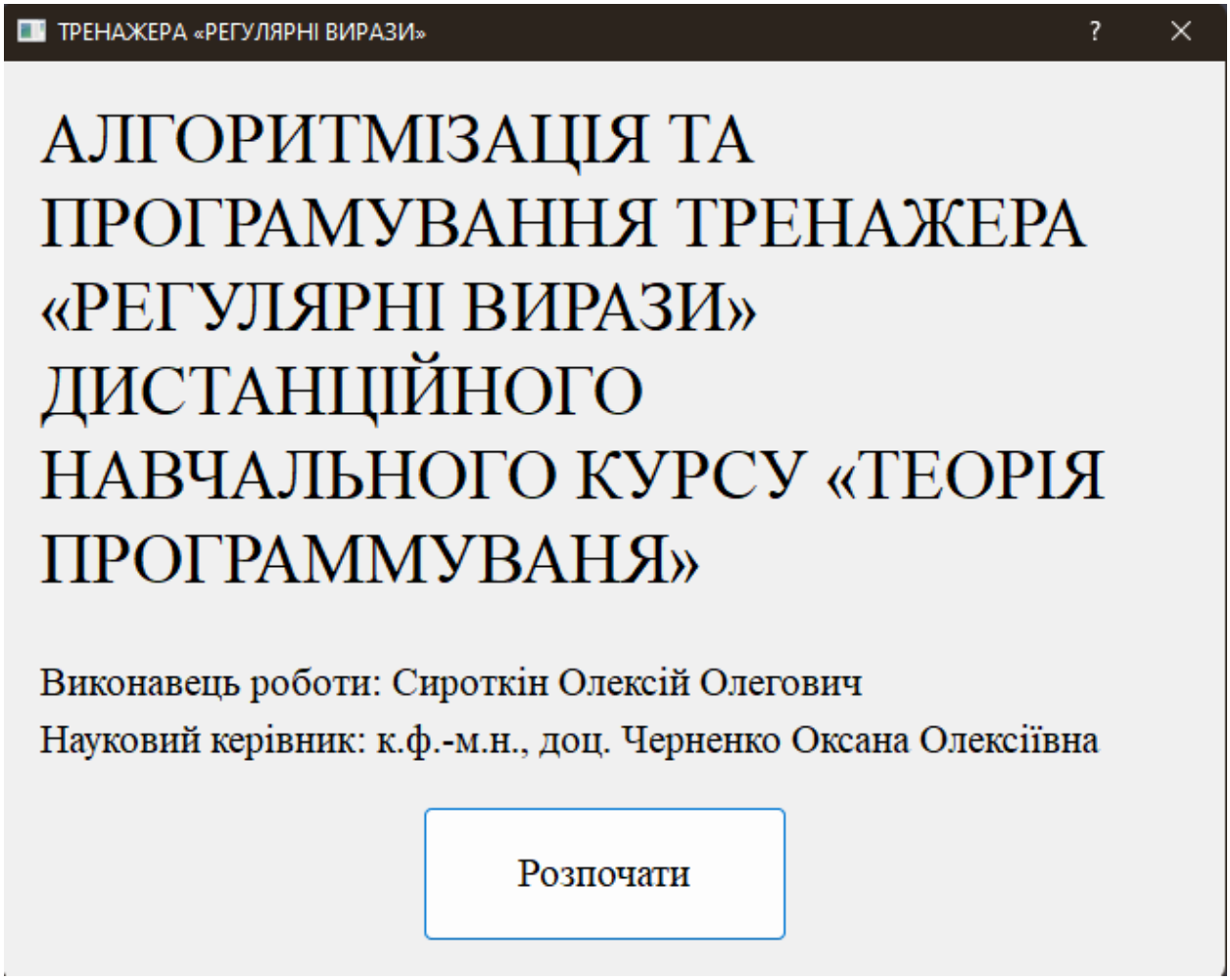



Рисунок 4.4 – Початкове меню

Після ознайомлення з темою та розробником тренажеру користувач переходить до першого практичного завдання, що має наступний вигляд:

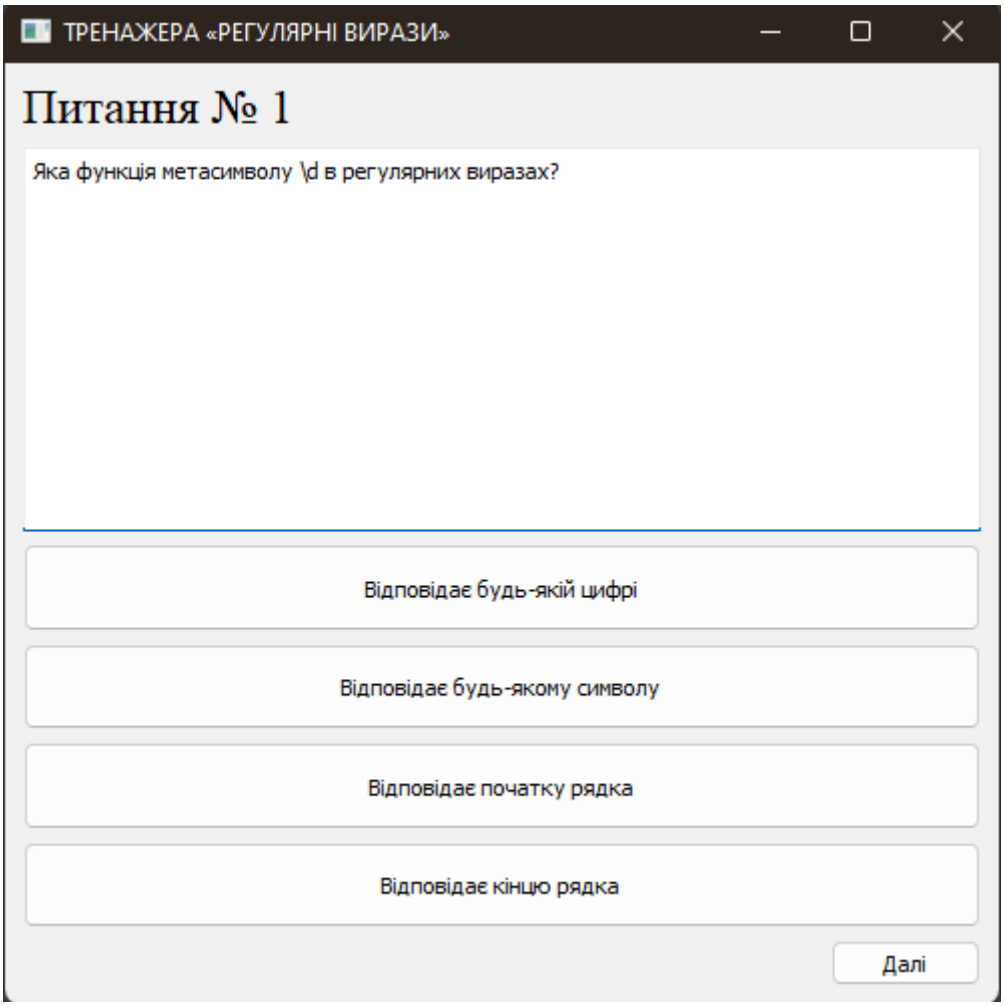



Рисунок 4.5 – Меню з умовою

Після вибору правильного варіанту кнопка світиться зеленим (див. Рисунок 4.6), у іншому випадку кнопка починає горіти червони (див. Рисунок 4.7).

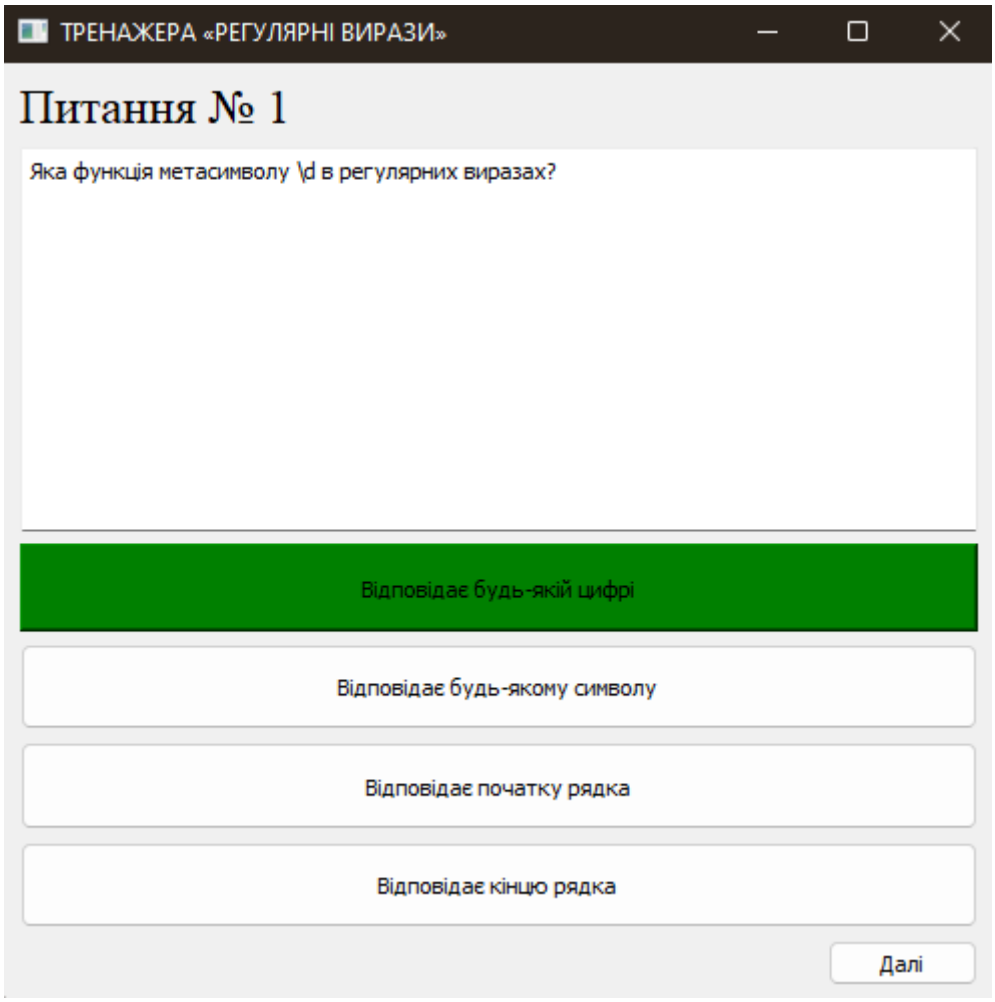



Рисунок 4.6 – Тренажер після вибору правильної відповіді

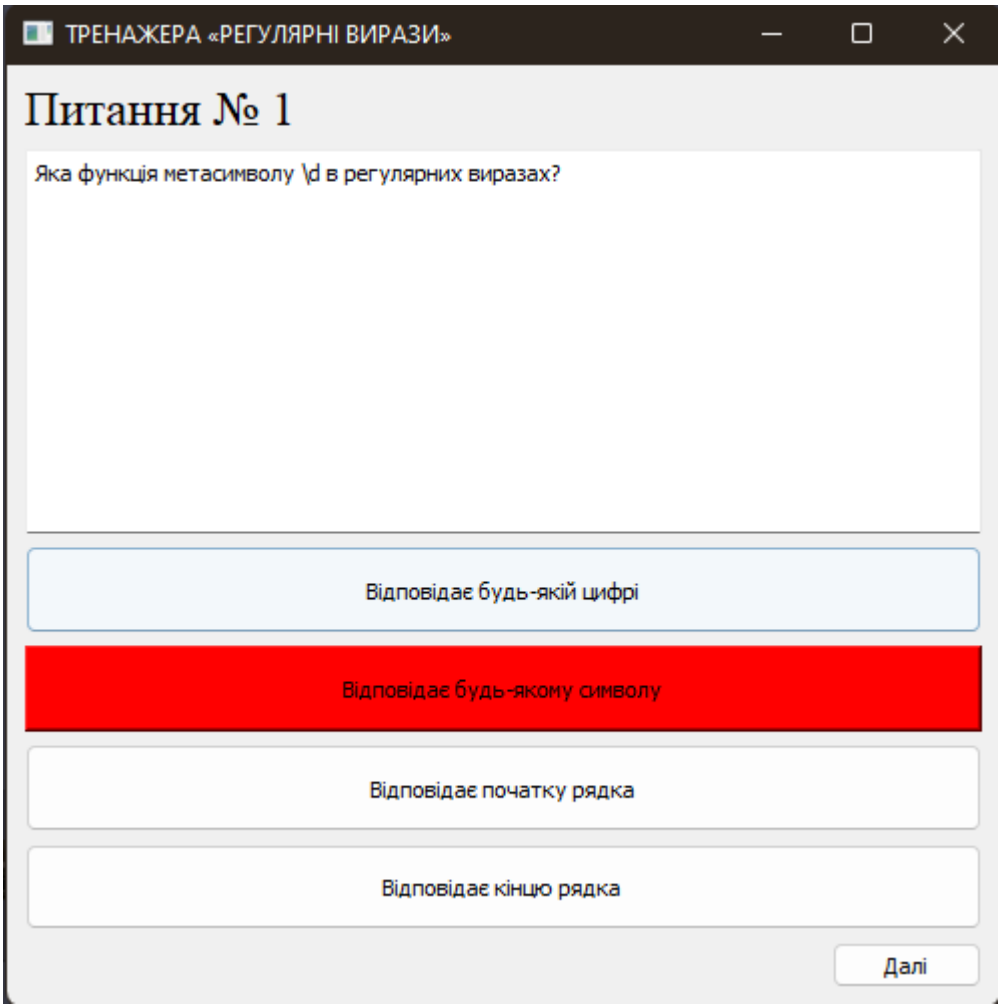
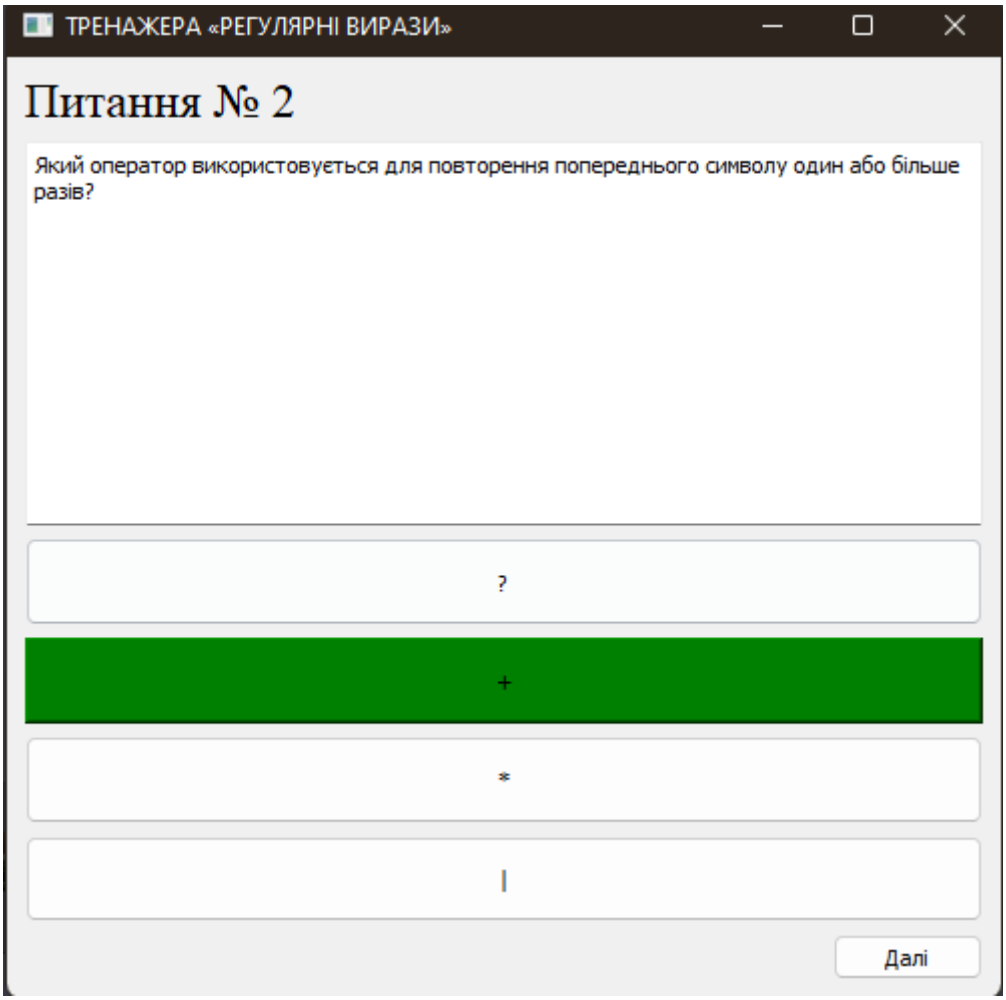



Рисунок 4.7 – Тренажер після вибору неправильної відповіді

Користувач, натиснувши на кнопку «Далі», переходить до наступного




практичного завдання.

Рисунок 4.8 – Тренажер після обрання правильної відповіді

Тренажер є універсальним. Разом з програмою ми маємо і файл конфігурації питань у форматі JSON.

```
[
    {
        "question": "Яка функція метасимволу \\d в регулярних виразах?",
        "answer": [
            "Відповідає будь-якій цифрі",
            "Відповідає будь-якому символу",
            "Відповідає початку рядка",
            "Відповідає кінцю рядка"
        ],
        "correct_answer": "Відповідає будь-якій цифрі"
    },
    {
        "question": "Який оператор використовується для повторення попереднього символу один або більше разів?",
        "answer": [
            "?",
            "+",
            "*",
            "|"
        ],
        "correct_answer": "+"
    },
    {
        "question": "Як в регулярних виразах позначається початок рядка?",
        "answer": [
            "^",
            "$",
            "*",
            "."
        ],
        "correct_answer": "^"
    },
    {
        "question": "Яка послідовність символів використовується для вибору одного з декількох символів?",
        "answer": [
            "()",
            "{}",
            "[]",
            "//"
        ],
        "correct_answer": "[]"
    },
```

Рисунок 4.9 Файл конфігурації запитань

Завдяки такій структури викладач може змінювати питання, змінювати правильну відповідь та самі варіанти відповідей. Структура JSON має такі поля як:

- Question
- Answer
- Correct_answer

Поле Question – це саме запитання яке буде виводитись на екран користувачу.

Поле Answer має массив з 4 відповідями. Завдяки цьому і з’являється відповіді для користувачу, які він може обрати.

Поле correct_answer – дане поле відповідає за правильну відповідь. Саме сюди викладач і пише правильну відповідь, яка повина співпадати з одним із варіантом відповіді. Якщо даний конфігураційний файл не правильно налаштувати, програма буде працювати некоректно, але якщо даного файлу взагалі немає, програма не запуститься.

Після завершення тренажера користувач переходить до перевірки свого розв'язку та має можливість скористатись кнопкою "Повторити", щоб пройти тренажер ще раз і закріпити навчальний матеріал для кращого засвоєння. (див. Рисунок 4.17 - 4.18).

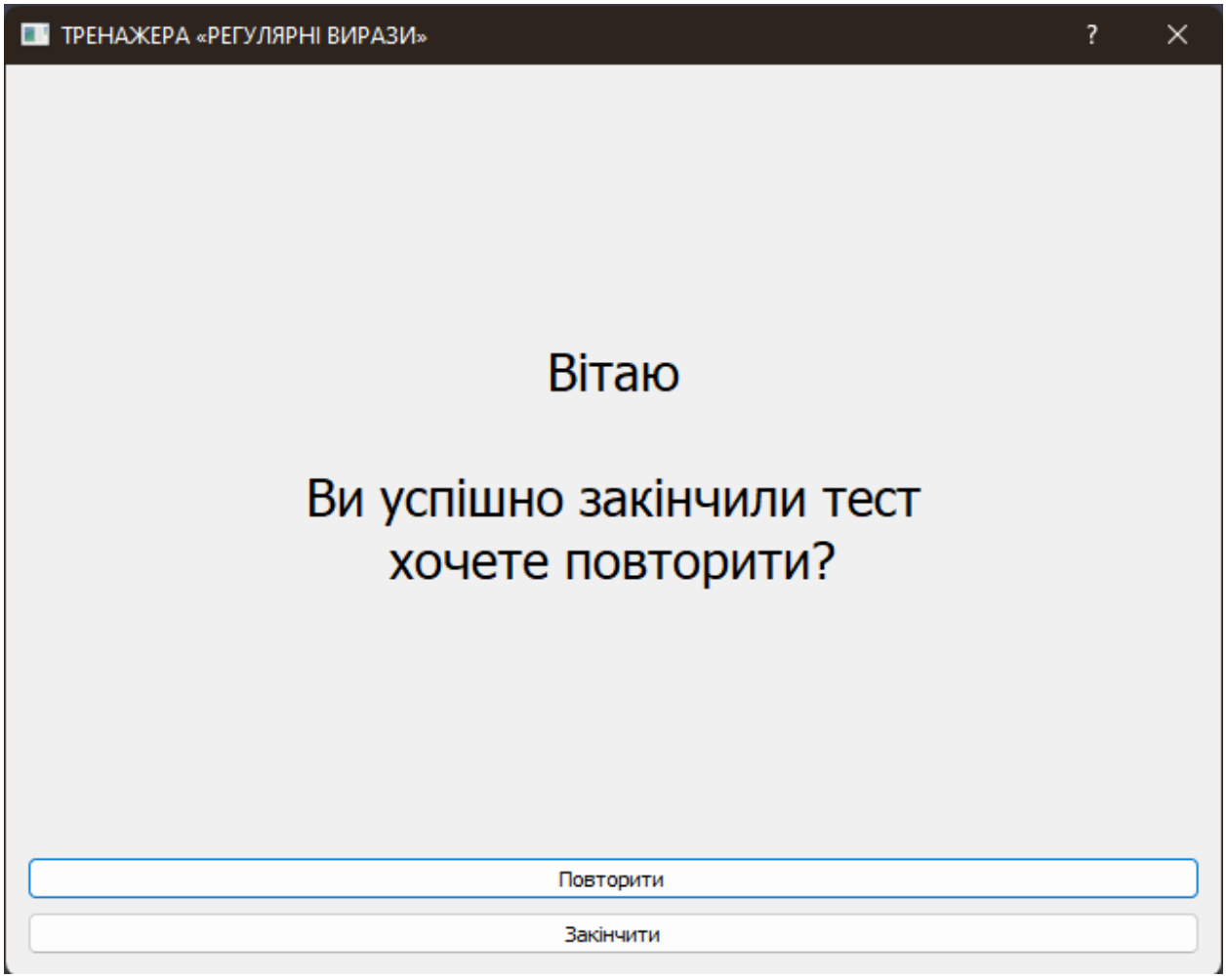



Рисунок 4.10 – Кнопка «Повтор» на фінальному вікні тренажеру

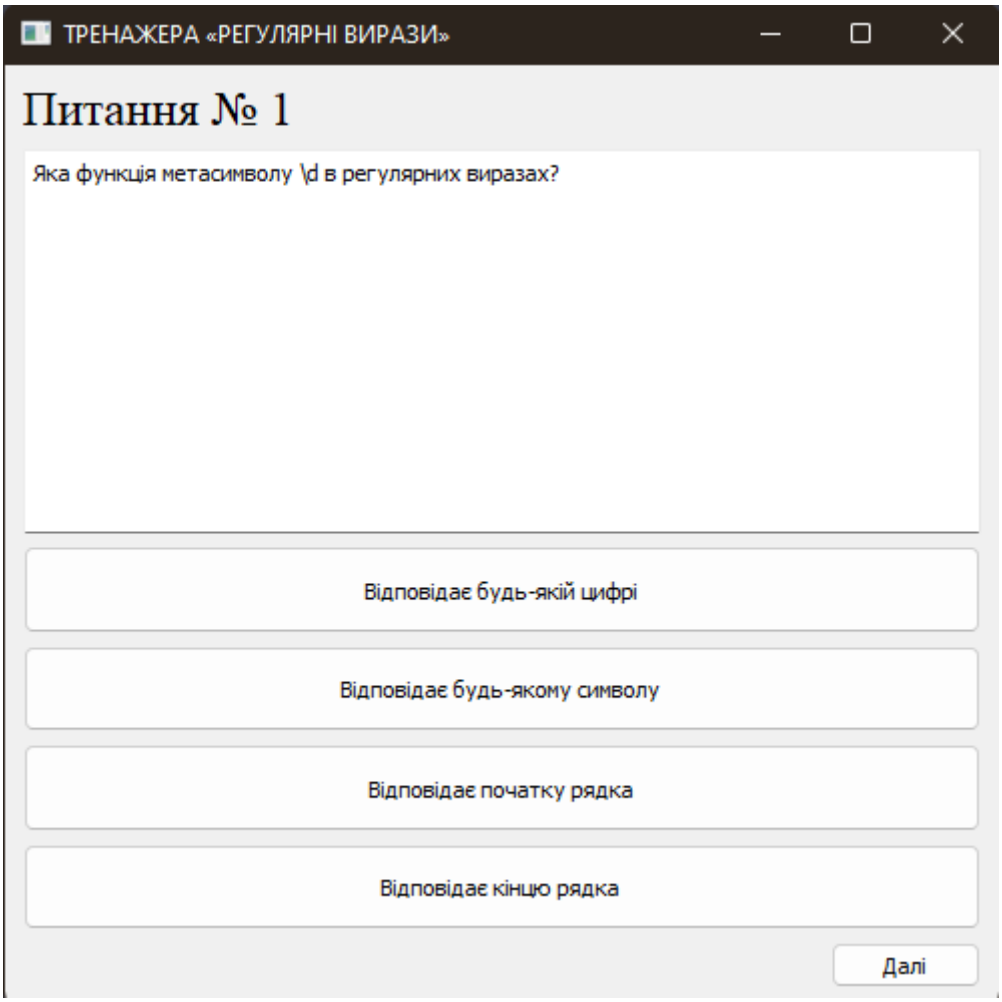



Рисунок 4.11 – Тренажер після натиснення на кнопку «Повтор» на фінальному вікні роботи з ним

# ВИСНОВОК

Під час створення кваліфікаційної роботи було опрацьовано тему та знайдено інформацію для теоретичної частини, створено алгоритм роботи тренажеру та блок-схеми цього алгоритму. Програмно реалізовано елементи тренажеру "Регулярні вирази" для дистанційного навчального курсу "Теорія програмування". Основними позитивними сторонами виконаної роботи та розробленого тренажеру являються:

- Складено алгоритм роботи навчального тренажеру;
- Створено блок-схеми в відповідності до розробленого алгоритму;
- Обрано платформу та середовище для розробки, а також мову для програмування навчального тренажеру згідно розробленого алгоритму;
- Реалізовано програму-тренажер відповідно на обраній платформі за допомогою обраної мови програмування,
- Створено зручний та приємний в використанні інтерфейс тренажеру,
- Забезпечено перевірку введеної відповіді, після вибору варіанту відповіді повідомити користувача про правильність обраного варіанту,
- Здійснено перевірку тренажера на валідність,
- Забезпечено роботу тренажера на операційній платформі Windows будь-якої версії,
- Створено інструкцію користувача для роботи з тренажером.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. О. В. Ольховська, О. О. Черненко МЕТОДИЧНІ РЕКОМЕНДАЦІЇ до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня бакалавра / О. В. Ольховська, О. О. Черненко – Полтава : РВВ ПУЕТ, 2022. – 68 с.
2. Документація Python 3.10.11 [Електронний русурс]. – Режим доступу до ресурсу: https://docs.python.org/3.10/
3. Python. Книга рецептов // Дэвид Бизли, Брайан К. Джонс – 2019– 650 ст
4. Документація PyQt5 [Електронний ресурс]. – Режим доступу до ресурсу: https://www.riverbankcomputing.com/static/Docs/PyQt5/
5. Python. Лучшие практики и инструменты // Яворски Михал, Зиаде Тарек – 2021 – 560 ст
6. Python [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python
7. Курс по теорії программування [Електронний курс] – Режим доступу до ресурсу: http://www2.el.puet.edu.ua/st/mod/page/view.php?id=158327
8. Python 3 и PyQt 5. Разработка приложений. // Дронов Владимир и Прохоренок Николай 2019г – 833 ст
9. Ерік Меттес, "Python. Курс загального вводу" / Ерік Меттес, 2019 - 560 сторінок.
10. Аль Свейгарт, "Автоматизація нудних справ з Python. Практичне програмування для абсолютних початківців" / Аль Свейгарт, 2019 - 504 сторінки.
11. Майкл Сіпсер, "Вступ до теорії обчислень" / Майкл Сіпсер, 2019 - 504 сторінки.
12. Майкл Т. Гудріч, Роберто Тамассія, Майкл Х. Голдвассер, "Структури даних і алгоритми на Python" / Майкл Т. Гудріч, Роберто Тамассія, Майкл Х. Голдвассер, 2019 - 735 сторінок.

13. Вес Маккінні, "Аналіз даних на Python. Робота з Pandas, NumPy і IPython" / Вес Маккінні, 2019 - 544 сторінки.
14. Аллен Дауні, "Думай Python. Як думати як комп'ютерний вчений" / Аллен Дауні, 2019 - 240 сторінок.
15. Девід Бізлі, Брайан К. Джонс, "Кулінарна книга Python" / Девід Бізлі, Брайан К. Джонс, 2020 - 706 сторінок.
16. Ден Бейдер, "Python Tricks. Буфет з приголомшливими функціями Python" / Ден Бейдер, 2019 (2-ге видання) - 304 сторінки.
17. Лючано Рамальо, "Плавний Python. Зрозумілий, лаконічний та ефективний програмування" / Лючано Рамальо, 2019 - 792 сторінки.
18. Джон Зелле, "Вступ до програмування. Python" / Джон Зелле, 2020 - 528 сторінок.

# ДОДАТОК А

```
import sys
import json

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QSizePolicy

from quest import Ui_MainWindow
from startWindow import Ui_StartWindow
from finalWindow import Ui_FinalWindow

class MyWindow(QtWidgets.QMainWindow):
    def __init__(self, perent=None):
        QtWidgets.QWidget.__init__(self, perent)
        self.ui = Ui_MainWindow()
        self.startWindow()
        self.__numberQuest = 0
        self.currentQuestion = []
        self.correctAnswer = ""
        self.__nextQuest = True
        self.questionLoaded()
        self.ui.setupUi(self)
        self.newQuestion()

        self.ui.answer_Button1.clicked.connect(self.clickedButton)
        self.ui.answer_Button2.clicked.connect(self.clickedButton)
        self.ui.answer_Button3.clicked.connect(self.clickedButton)
        self.ui.answer_Button4.clicked.connect(self.clickedButton)
        self.ui.next_Button.clicked.connect(self.newQuestion)

    def questionLoaded(self):
        with open('quest.json', 'r', encoding='utf8') as file:
            self.jsonQuestion = json.load(file)

    def newQuestion(self):
        if not self.__nextQuest:
```

```
            return

        if self.__numberQuest == len(self.jsonQuestion):
            self.ui.MainWindow.hide()
            self.finishWindow()
            return

        self.__nextQuest = False
        self.__clearStyleButton()

        self.currentQuestion = self.jsonQuestion[self.__numberQuest]
        buttonOne = self.currentQuestion["answer"][0]
        buttonTwo = self.currentQuestion["answer"][1]
        buttonThree = self.currentQuestion["answer"][2]
        buttonFour = self.currentQuestion["answer"][3]
        self.correctAnswer = self.currentQuestion["correct_answer"]

        self.ui.numberQuestion.setText(f"Питання № {self.__numberQuest+1}")
        self.ui.questionEdit.setPlainText(self.currentQuestion["question"])
        self.ui.answer_Button1.setText(buttonOne)
        self.ui.answer_Button2.setText(buttonTwo)
        self.ui.answer_Button3.setText(buttonThree)
        self.ui.answer_Button4.setText(buttonFour)

        self.__numberQuest += 1

    def clickedButton(self):
        button = self.sender()

        self.__clearStyleButton()

        if button.text() == self.correctAnswer:
            button.setStyleSheet("background-color: green;")
            self.__nextQuest = True
        else:
            self.__nextQuest = False
            button.setStyleSheet("background-color: red;")
```

```
    def __clearStyleButton(self):
        self.ui.answer_Button1.setStyleSheet("")
        self.ui.answer_Button2.setStyleSheet("")
        self.ui.answer_Button3.setStyleSheet("")
        self.ui.answer_Button4.setStyleSheet("")

    def finishWindow(self):
        finishWin = Ui_FinalWindow()
        finishWin.setupUi(finishWin)
        choise = finishWin.exec()

        if choise:
            self.ui.MainWindow.setVisible(True)
            self.__numberQuest = 0
            self.newQuestion()
        else:
            return


    def startWindow(self):
        startWin = Ui_StartWindow()
        startWin.setupUi(startWin)
        startWin.exec()


if __name__=='__main__':
        app = QtWidgets.QApplication(sys.argv)
        myapp = MyWindow()
        myapp.show()
        app.exec_()
        sys.exit()
```