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
«__»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«АЛГОРИТМІЗАЦІЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ТЕМИ «ЦИКЛИ МОВИ C#»

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

Виконавець роботи Сичик Олексій Сергійович
_____«__»_____202_р.
(підпис)
Науковий керівник к.ф.-м.н.,доц., Чілікіна Тетяна Василівна
_____«__»_____202_р.
(підпис)
Рецензент

ПОЛТАВА 2023 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри_____Олена ОЛЬХОВСЬКА
«____» _____ 202_ р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Алгоритмізація та розробка програмного забезпечення з теми «Цикли мови C#»
зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Сичик Олексій Сергійович
Затверджена наказом ректора №8-Н від «16» січня 2023 р.
Термін подання студентом роботи «___»_____202_ р.
Вихідні дані до кваліфікаційної роботи: методичні матеріали за темою роботи; «Алгоритмізація та розробка програмного забезпечення з теми «Цикли мови C#» стандарти, а також умови та алгоритми задач, з яких буде складатися програмне забезпечення.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
I. ПОСТАНОВКА ЗАДАЧІ
II. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1. Типи комп'ютерних симуляторів
2.2. Огляд середовищ та мов програмування для розробки тренажерів
III. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Огляд матеріалу з теми
3.2. Обґрунтування вибору програмних засобів для реалізації
3.3. Блок-схема програмного забезпечення
IV. ПРАКТИЧНА ЧАСТИНА
4.1. Опис програмної реалізації
4.2. Інструкція по роботі з програмним забезпеченням
4.3. Тестування симулятору
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А
Перелік графічного матеріалу: 3-4 аркуші блок-схем, інші необхідні ілюстрації.

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Чілікіна Т.В.		
Інформаційний огляд	Чілікіна Т.В.		
Теоретична частина	Чілікіна Т.В.		
Практична частина	Чілікіна Т.В.		

Календарний графік виконання дипломної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій, 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.
Здобувач вищої освіти_____Олексій СИЧИК
(підпис)
Науковий керівник _____ к.ф.-м.н., доц. Тетяна ЧІЛІКІНА
(підпис)

Результати захисту кваліфікаційної роботи
Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)
Протокол засідання ЕК №_____від «____»_____202_ р.
Секретар ЕК_____ _____
(підпис) (ініціал та прізвище)

Затверджую
Зав. кафедрою_____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«__»_____202_ р.

Погоджено
Науковий керівник_____
к.ф.-м.н. Тетяна ЧІЛІКІНА
«__»_____202_ р.

План

кваліфікаційної роботи ступеня магістр
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
Сичик Олексій Сергійович
Прізвище, ім'я, по батькові

на тему «Алгоритмізація та розробка програмного забезпечення з теми «Цикли мови C#»

Здобувач вищої освіти_____Олексій СИЧИК
(підпис)
«____»_____202__р.

АНОТАЦІЯ

Записка: 65 с., 11 рис., 1 додаток, 12 джерел.

Мета роботи – проектування та розробка елементів програмного забезпечення з теми «Алгоритмізація та розробка програмного забезпечення з теми «Цикли мови C#».

Об’єкт розробки – створення елементів програмного забезпечення у вигляді симулятора з теми «Алгоритмізація та розробка програмного забезпечення з теми «Цикли мови C#».

Методи дослідження та інформаційне забезпечення – використання програмного середовища MS Visual Studio, мова програмування C#.

Елементи симулятору розроблено на основі віконної програми з використанням системних бібліотек C# та платформи .NET Framework.

Результати дослідження. Виконано огляд поняття програми-симулятора, його видів та особливостей використання. Розроблено алгоритм програмного забезпечення «Цикли мови C#», побудовано його блок-схему. Виконана програмна реалізація симулятору в програмному середовищі MS Visual Studio мовою програмування C#.

Рекомендації щодо використання результатів дослідження. Програмне забезпечення симулятору можна використовувати для самостійного опрацювання відповідних тем дисципліни здобувачами освіти за спеціальністю «Комп’ютерні науки».

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
Симулятор	Програма-вчитель створена для вивчення і закріплення теоретичного матеріалу.
ДК	Дистанційні курси.
C#	Мова програмування.
.NET Framework	Платформа для створення звичайних застосунків.
IDE MS Visual Studio	Інтегроване середовище програмування.
ЕОМ	Електронно-обчислювальна машина.
WF	Windows Forms.

ВСТУП

Проблему впровадження інформаційних технологій в навчальний процес досліджують у великій кількості робіт. В останні роки створено непогані варіанти симуляторів широкого кола спеціальностей проте вони не узгоджені один з одним за більшістю параметрів, відрізняються операційними системами, способом подання матеріалу, їх зміст не дозволяє використовувати їх в межах єдиної освітньої програми.

Сучасні інформаційні та комп'ютерні технології, передовсім електронні програми-симулятори, широко використовуються для навчання. Водночас комп'ютерні технології дають змогу не тільки зменшити вартість навчання, але й покращити його якість. Адже відомо, що інформаційні технології навчання сприяють розвитку особистості студента, підготовці його до самостійної продуктивної діяльності в умовах інформаційного суспільства.

Вони передбачають (крім передачі інформації і закладених у ній знань):

- інтелектуальний розвиток (конструктивне, алгоритмічне мислення, завдяки особливостям спілкування з комп'ютером;
- креативний розвиток (творче мислення) за рахунок зменшення частки репродуктивної діяльності;
- професійний розвиток (формування уміння приймати оптимальні професійні рішення у складних ситуаціях під час комп'ютерних ділових ігор і роботи з програмами-тренажерами).

Актуальність роботи – за допомогою таких симуляторів можна підійняти успішність студентів тому, що їх використання не потребує постійного нагляду та перевірки викладача. Матеріал можна проходити по декілька разів щоб краще засвоїти матеріал.

Мета кваліфікаційної роботи – розробка елементів програмного забезпечення з теми «Алгоритмізація та розробка програмного забезпечення з теми «Цикли мови C#»

Об'єктом розробки є процес створення програмного забезпечення у вигляді тренажера з теми «Цикли мови C#» як складової дистанційного навчання інформаційних дисциплін.

Предмет розробки —програма-симулятор з теми « «Цикли мови C#».

Кваліфікаційна робота складається з чотирьох розділів. У першому розділі розглянуто постановку задачі. У другому розділі розглянуто типи комп'ютерних тренажерів, системи, середовища програмування, середовища для розробки програмного забезпечення. У третьому розділі представлено огляд матеріалу з теми, алгоритм робити програмного забезпечення. Четвертий розділ присвячений опису програмної реалізації та інструкції користування програмним забезпеченням.

Результатом виконання кваліфікаційної роботи є розробка програмного забезпечення у вигляді програми-симулятору на мові програмування C# у середовищі MS Visual Studio.

Обсяг пояснювальної записки: 65 стор., в т.ч. основна частина 32 стор., додатки - 21 стор., джерел – 12 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Рішення будь-якої прикладної задачі з використанням електронних обчислювальних машин (ЕОМ) складається з наступних етапів:

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

Постановка завдання розробки програмного забезпечення є найпершим і найбільш важливим етапом при проектуванні програмного забезпечення. Саме тут закладається фундамент майбутньої програми. Якщо на цьому етапі допущені помилки або не передбачені якісь важливі деталі, то це відіб'ється на кінцевому результаті, а вносити зміни і виправлення в такий проект буде вкрай проблематично. Більш того, якщо завдання поставлене невірно, то її подальше рішення не має сенсу. На етапі постановки завдання необхідно відповісти на наступні питання:

- чи існують методи або способи вирішення завдання без використання ЕОМ, чи потрібно вирішувати це завдання на ЕОМ;
- які «дивіденди» принесе використання ЕОМ для вирішення завдання, що буде покращено, прискорене, оптимізовано, зекономлено при використанні ЕОМ, яка основна мета застосування ЕОМ;
- що необхідно з обладнання, крім універсальної ЕОМ, який тип ЕОМ необхідний для вирішення завдання, яка платформа та ЕОМ може підійти, яка операційна система буде керувати роботою ЕОМ, яке додаткове програмне забезпечення потрібно;
- які бізнес-процеси і документообіг прикладної предметної області, де буде застосовуватися розробляється програмне забезпечення;
- який формат вихідних (вхідних) даних, які дані і в якій формі необхідні для вирішення завдання;

- який формат проміжних і вихідних даних, які дані і в якій формі необхідно отримати;
- який інтерфейс користувача програмного забезпечення потрібно забезпечити, який інтерфейс з додатковим обладнанням необхідний;
- яка «глибина» опрацювання користувальницької, інженерної, програмної і конструкторської документації, експлуатаційних документів, методичних посібників і керівництв по використанню програми, хто і як буде застосовувати результати рішення.

Перший крок у проектуванні програмного забезпечення полягає в точному формулюванні цілей впровадження програми. Необхідно, щоб цей процес був гнучким та організованим і працював протягом тривалого часу, оскільки на будь-якому етапі розробки або впровадження можуть розкриватися раніше непередбачені проблеми, які потребують внесення в проект певних змін. Необхідно також заздалегідь визначити умови внесення несуттєвих змін в проект. Всі домовленості такого плану повинні оформлятися розробником і замовником програми в офіційному порядку у вигляді окремих пунктів у договорі на розробку програмного забезпечення, додаткових протоколів чи актів.

З офіційним висновком договору зазвичай передають:

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

Після завершення етапу попередніх досліджень складають список вимог, що пред'являються 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).

Головна мета на етапі аналізу формальної постановки задачі і складання вимог до програми полягає в пошуку і поданні таких ситуацій, які можуть привести до збою в роботі програми і у визначенні причин і способів подолання таких ситуацій.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Типи комп’ютерних симуляторів

Дистанційне навчання є важливим інструментом для забезпечення доступу до освіти у будь-який час та у будь-якому місці. Симулятори в цьому контексті виступають як невід'ємна складова, спрямована на створення ефективного навчального середовища.

Мета створення навчальних симуляторів для ДК призначена для підвищення якості та ефективності навчання студентів, шляхом віртуального відтворення різних сценаріїв та завдань. Завдяки навчальним симуляторам студенти мають змогу отримувати навички у віртуальному середовищі, що відображають реальні умови де саме ці навички можуть знадобитися в подальшому. Симулятори також сприяють інтерактивності та індивідуалізації навчання, дозволяючи студентам вчитися власним темпом та вирішувати завдання, що відповідають їх особистим потребам.

Саме таким чином стає можливим підвищити мотивацію та розширити спектр засобів навчання та заохочення студентів здобувати більше знань і спрощення виконання ними завдань.

Можна виділити наступні типи комп'ютерних симуляторів:

1. Симулятор електронного екзамінатора. Це простий програмний продукт, що реалізовується на всіх видах вітчизняної і зарубіжної обчислювальної техніки. Основна його функція – це заміна живого екзаменатора в строго регламентованих областях (техніка безпеки різних виробництв, правила дорожнього руху і тому подібне). Як правило, такий екзаменатор містить набір квитків з декількох питань, пропонованих що іспитується у випадковому порядку, і ряду неправильних і однієї правильної відповіді на кожне питання. Залежно від складності екзаменатора, він може забезпечувати також наступні можливості:

- показ малюнків в кадрі питання;
- показ мультфільмів (анімація) в кадрі питання;
- аналіз відповіді що іспитується у вигляді чисел і формул;

- попереднє навчання (показ правильних відповідей);
- редагування старих і створення нових питань.

Вартість розробки подібних екзаменаторів найнижча (прості тренажери такого класу може написати школяр).

2. Статичні (або логіко-динамічні) симулятори. Основна особливість полягає в тому, що в таких програмах відсутня фізико-математична модель процесів, що відбуваються в устаткуванні, але показується і перевіряється певний порядок дій. Порядок дій зазвичай жорстко задається; у складніших випадках передбачаються розгалуження в ланцюжку дій, що забезпечується логічними функціями (логіко-динаміна модель). Головні недоліки:

- неможливість відхилення навчаного від скільки завгодно складною, але все одно жорстко заданого ланцюжка дій;
- трудність програмування динамічних ефектів (навіть простої зміни свідчень приладів).

3. Динамічні симулятори. Мають в своїй основі математичну модель реальних фізичних процесів і тому найбільш корисні для якісного навчання персоналу. Зрозуміло те, що для розрахунку достатньо повної моделі потрібно вельми значні машинні ресурси. Тому модель слід спростити, але так, щоб її поведінка в обумовлених технічним завданням рамках відповідало реальній системі з певною точністю. Це складне творче завдання для інженера і науковця. Можна з упевненістю сказати, що вартість розробки такого наукового продукту в багато разів перевищує ціну, яку згоден сплатити замовник. Так пояснюється відносно невеликою кількістю розробників тренажерів такого класу і відчутна ніша на ринку подібних продуктів. Вартість розробки: висока.

Зупинимося докладніше на динамічних симуляторах, як найцікавіших і складніших програмних продуктах. Є два різні способи їх створення. Перший полягає в написанні окремої програми для кожного окремого тренажера, другої, - у використанні спеціального інструменту розробника, який дозволяє в багато разів прискорити розробку. У першому випадку можливе досягнення красивих спеціальних ефектів, але дуже утруднена модифікація тренажера. Як правило, програми цього класу дуже прості, тому такі локальні тренажери поширеніші.

При використанні конструктора розробником тренажера повинні бути не програміст, а технолог, що володіє апаратом прикладної математики. Загальна властивість більшості конструкторів - складання динамічної моделі з "блоків" - стандартних елементів, що описують певні об'єкти управління (або прилади) і стандартні математичні операції, передавальні функції, логіку. Конструктори рідко і не дуже охоче пропонуються на продаж по причинах дуже високої вартості розробки і малого тиражу. З іншого боку, при перспективному плануванні комп'ютеризації підготовки персоналу має сенс піти на підвищені витрати, щоб має можливість редагувати наявні моделі і створювати нові за власним розсудом.

При виборі конструктора слід враховувати:

- набір стандартних елементів і можливість його розширення;
- можливі режими роботи і розв'язування завдання (показ правильних дій, сценарії аварійних ситуацій, оцінка навчаного);
- якість і ергономічні характеристики інтерфейсу (зручність роботи) розробника і навчаного.

Ряд колективів використовують (і пропонують на продаж) конструктори динамічних симуляторів більш менш прийнятної якості. Наш конструктор виконаний за об'єктно-орієнтованою технологією і забезпечує зручний графічний інтерфейс як для навчання, так і для розроблення. Моделі складаються із стандартних об'єктів, кожен з яких має своє зображення, механізм управління і спосіб моделювання. Набір елементів за бажанням замовника легко може бути розширений. Закінчена модель є набором з декількох вікон (вікна користувача), об'єднаних один з одним за ієрархічним принципом, або за принципом циклічного списку. У кожен окремий момент на екрані може бути видно одне або декілька вікон. Частину вікон з керованими об'єктами складають інтерфейс навчання. Управління моделлю так само легко здійснюється як за допомогою миші, так і з клавіатури. У недоступних для навчання вікнах розробник збирає динамічну модель, маючи для цього багатий набір стандартних елементів.

2.2. Огляд середовищ та мов програмування для розробки тренажерів

Існує досить багато середовищ для розробки програмного забезпечення, наприклад такі як:

- ***NetBeans*** **–** інтегроване середовище розробки, що підтримує ряд мов програмування, таких як Java, PHP, C++, HTML та інші.
- ***Unity*** – для розробки ігор та інтерактивних додатків, використовує мови програмування C# та JavaScript.
- ***PyCharm*** – Спеціалізоване середовище для розробки проектів на мові програмування Python.
- ***Visual Studio*** – використовується для розробки програм під управлінням операційних систем Windows, а Visual Studio Code - для різноманітних мов програмування та платформ.

Виходячи з прикладів абсолютно будь-які середовища варіюються в залежності від мов програмування, типів проекту та індивідуальних потреб розробників.

Хочу відмітити декілька сильних сторін які були використанні при розробці даного програмного забезпечення.

По-перше, мова програмування C# – об'єктно-орієнтована мова програмування з безпечною системою типізації. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників - мов C++, Object Pascal, Модула і Smalltalk - C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, множинне спадкування класів (на відміну від C++).

По-друге, середовище Microsoft .NET Framework – це технологія,

впроваджена корпорацією Майкрософт як платформа для створення звичайних програм для персональних комп'ютерів, а також веб-додатків або порталів. Однією з ідей .NET є створення сумісності служб, написаних на різних мовах програмування. Ця функція впроваджується корпорацією Майкрософт як перевага для .NET виконання XML-додатків і веб-служб.

Платформа .NET Framework – це технологія, яка підтримує створення додатків.

Основою для .NET Framework є середовище CLR. Середовище виконання може розглядатися як агент, який керує кодом під час виконання програми і надає основні послуги. Створення умов суворої типізації та інших видів контролю правильності коду, які забезпечують безпеку і надійність при роботі. Основним завданням виконання середовища є управління кодом. Бібліотека класів – це повна об'єктно-орієнтована колекція, що дозволяє повторно використовувати типи, для розробки програм.

По-третє, середовище розробки Microsoft Visual Studio – один з продуктів компанії Майкрософт, який представляє з себе інтегровану середу розробки програмного забезпечення і ряд інших інструментальних засобів. Даний продукт дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб- служби як в не керованому, так і керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET, Framework .NET Compact Framework і Microsoft Silverlight

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Решта вбудовуваних інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Огляд матеріалу з теми

Цикл – це різновид керуючої конструкції у високорівневих мовах програмування, призначений для організації багаторазового виконання набору

інструкцій. Також циклом може називатися будь-

яка послідовність інструкцій,

що багаторазово виконується, організована будь-

яким способом (наприклад, за

допомогою умовного переходу).

Послідовність інструкцій, призначена для багаторазового виконання, нази вається тілом циклу. Одиничне виконання тіла циклу називається ітерацією. Ви раз визначальний, чи в черговий раз виконуватиметься ітерація, чи цикл заверш иться, називається умовою виходу чи умовою закінчення циклу (або умовою пр одовження в залежності від того, як інтерпретується його істинність – як ознака необхідності завершення або продовження циклу). Мінлива, що зберігає поточний номер ітерації, називається лічильником ітерацій циклу або п росто лічильником циклу. Цикл не обов'язково містить лічильник, лічильник не повинен бути один

умова виходу з циклу може залежати від декількох змінних в циклі змінних, а м оже визначатися зовнішніми умовами (наприклад, настанням певного часу), в о станньому випадку лічильник може взагалі НЕ знадобитися.

Безумовні цикли. Іноді у програмах використовуються цикли, вихід із яких не передбачений логікою програми. Такі цикли називаються безумовними, або нескінченними. Спеціальних синтаксичних засобів для створення нескінчен них циклів, зважачи на їх нетиповість, мови програмування не передбачають,

тому такі цикли створюються з допомогою конструкцій, призначених для створення звичайних (або умовних) циклів. Для забезпечення нескінченного по вторення перевірка умови в такому циклі або відсутня (якщо дозволяє синтаксис, як, наприклад, у циклі LOOP … END LOOP мови Ада), або замінюється конс тантним значенням

(поки правда робити … в Паскалі). В мові С використовується

цикл для (;;) з не заповненими секціями.

Цикл з передумовою – цикл який виконується поки деяка умова є істиною, вказана перед його початком. Ця умова перевіряється до виконання тіла циклу, тому тіло може бути НЕ виконано ні разу (якщо його умова з самого початку була хибною). У більшості процедурних мов програмування реалізується оператором **while**.

Приклад циклу while:

while (<умова>)

{

<тіло>;

};

Цикл з постумовою – цикл в якому умова перевіряється після виколнання тіла циклу. Звідси випливає, що тіло завжди виконується хоча б один раз. Реалізується оператором **do/while**.

Приклад циклу do/while:

do

{

    <тіло>;

}

while (<умова>);

Цикл з лічільником – це цикл, у якому деяка змінна змінює своє значення від заданого початкового до кінцевого з деяким кроком, і для кожного значення цієї змінної виконується один раз. Реалізовується оператором **for**.

Приклад циклу for:

for (<ініціалізація>; <умова>; <порядок виконання>)

{

    <тіло циклу>;

}

Де, **ініціалізація** – встановлення початкових параметрів лічільника, **умова**

– як тільки умова буде виконана повністю – відбудеться вихад з циклу, **порядок виконання** – команда збільшення або зменшення лічільника.

Спільний цикл – цикл який виконує деяку операцію для об’єктів із заданої множини, без явної вказівки порядку перерахування цих об’єктів. Довільність дає можливість оптимізації виконання циклу за рахунок організіції доступу НЕ в заданому порядку програмістом, а в найбільш вигідному порядку. Реалізовується оператором **foreach**, тобто на що нам і вказує його назва “для кожного”.

Приклад циклу foreach:

foreach (<тип змінної> <назва змінної> in <назва массиву(множини)>)

{

    <тіло циклу>;

}

Де, **тип змінної** – тип даних масиву(множини) який потрібно “перебрати”, **назва змінної** – звичайне ім’я за допомогою якого можливо звернутися до елементів масиву, **назва масиву** – назва самої множини яку потрібно “перебрати”.

3.2. Обґрунтування вибору програмних засобів для реалізації

Вибір середовища розробки MS Visual Studio та Windows Forms для реалізації програмного продукту може бути обґрунтований декількома факторами ось декілька з них:

- Швидка розробка інтерфейсу користувача: Windows Forms надає можливість швидко створювати інтерфейс користувача за допомогою графічного дизайнера в Visual Studio. Це дозволяє розробникам швидко пристосовуватися до змін у вимогах та легко взаємодіяти з дизайнерами та іншими членами команди.
- Широкий вибір елементів керування: Windows Forms включає в себе багато вбудованих елементів керування, які можна легко використовувати для побудови різноманітних інтерфейсів

користувача. Це включає в себе кнопки, тексти, таблиці, списки та інші елементи.

- Windows Forms, як правило, використовується для розробки десктоп-додатків з використанням мови програмування C#. C# - це мова, яка має чіткий синтаксис, велику кількість вбудованих бібліотек та відмінну підтримку об'єктно-орієнтованого програмування.
- Широка підтримка .NET Framework: Windows Forms базується на технології .NET Framework, яка надає потужний фреймворк для розробки десктоп-додатків. .NET Framework включає багато корисних бібліотек та компонентів, які спрощують розробку та покращують продуктивність.

Загалом, використання Visual Studio та Windows Forms дозволяє розробникам створювати ефективні та швидкі десктоп-додатки для операційної системи Windows з надійними інструментами та підтримкою .NET Framework.

На стартовому екрані (див. Рисунок 4.1) виводиться наступна інформація:

- привітання;
- можливості даного тренажеру;
- кнопки для початку проходження практичних завдань та теорії.

При натисканні кнопки *«Загальні відомості»*, відобразиться інформація з теми про цикли на мові C#, де користувач може ознайомитися з циклами їх використанням, написання синтаксису.

Якщо на боковій панелі натиснути будь-яку кнопку з категорії *«Практична частина»*, то тут користувач може розпочати виконувати практичні завдання знання яких він засвоїв читаючи теорію:

Кожна практична частина з кожного циклу має певну кількість завдань на проходження яких не має обмеження у часі:

- Практична частина с циклу **for** має 6 завдань.

- Практична частина с циклу **foreach** має 5 завдань.
- Практична частина с циклу **while/do-while** має 5 завдань.

Також є 2 типи практичних завдань:

- правильність написання синтаксису;
- надання відкритої відповіді, що буде виведене в результаті виконання деякого блоку коду.

1 крок. При запуску програми користувач може почати працювати як з теоретичною частиною де він може ознайомитися з тим, що таке цикли, застосування та приклади їх синаксичного написання у коді так і з практичною, якщо у нього є деякі навички роботи з циклами.

2 крок. Користувач починає проходити деяку практичну частину та пише відповіді в пусті поля. При неправильно написаних відповідях вказується помилка – обведення місця для вводу червоним кольором. При правильному – обведення місця для вводу зеленим кольором.

3 крок. Після того як користувач написав відповіді він натискає кнопку "*Підтвердити відповідь*". У разі якщо все ж таки десь була допущена помилка з'явиться повідомлення про помилку і поля де вона була допущена будуть обведені червоним кольором, але якщо усі відповіді були коректні, то після підтвердження відповіді користувач зможе перейти до наступного завдання натиснувши на кнопку *«Наступне»*.

Кінцевий крок. Після того як користувач завершив проходження будь-якої практичної частини на екран з'явиться наступне повідомлення з текстом про проходження практичної частини з теми про цикли: **for**, **foreach** або **while/do-while**.

Приклад роботи з навчальним симулятором описано наступними кроками:

1 крок. Після того як користувач ознайомився з теоретичним матеріалом він може приступити до роботи з практичним матеріалом. Обирає будь-яку практичну частину з запропонованих у лівому меню.

2 крок. Користувач отримує практичне завдання:

Практичне завдання №1 з циклу for. Використайте цикл for та виведіть

змінну «i» 10 разів.

Заповнити пропуски (“…”) у коді.

```
for (int …; i < …; i++)
{
   Console.WriteLine(i);
}
```

Правильні відповіді:

1) i = 0;
2) 10.

3 крок. Натиснути кнопку *«Підтвердити відповідь»*. У разі якщо десь була допущена помилка з’явиться повідомлення про помилку,і поля, де вона була допущена будуть обведені червоним кольором, але якщо усі відповіді були коректні, то після підтвердження відповіді користувач зможе перейти до наступного завдання натиснувши на кнопку *«Наступне»*.

4 крок. Користувач отримує наступне завдання.

Практичне завдання №2 з циклу for. Зупиніть цикл коли змінна «i» буде дорівнювати «5».

```
for (int i = 0; i < 10; i++)
{
    if (i == 5)
    {
        (…);
    }
}
Console.WriteLine(i);
```

Правильні відповіді:

1) break.

5 крок. Натиснути кнопку *«Підтвердити відповідь»*. У разі якщо десь була допущена помилка з’явиться повідомлення про помилку,і поля, де вона була допущена будуть обведені червоним кольором, але якщо усі

відповіді були коректні, то після підтвердження відповіді користувач зможе перейти до наступного завдання натиснувши на кнопку *«Наступне».*

6 крок. Користувач отримує наступне завдання.

Практичне завдання №3 з циклу for. У наступному циклі, коли значення «i» буде дорівнювати «3» перейдіть безпосередньо до наступного значення.

```
for (int i = 0; i < 10; i++)
{
    if (i == 3)
    {
        (…);
    }
}
Console.WriteLine(i);
```

Правильні відповіді:

1) countinue.

7 крок. Натиснути кнопку *«Підтвердити відповідь»*. У разі якщо десь була допущена помилка з’явиться повідомлення про помилку,і поля, де вона була допущена будуть обведені червоним кольором, але якщо усі відповіді були коректні, то після підтвердження відповіді користувач зможе перейти до наступного завдання натиснувши на кнопку *«Наступне».*

8 крок. Користувач отримує наступне завдання.

Практичне завдання №4 з циклу for. Запишіть умову циклу так, щоб він віднімав від себе кожного разу значення «2» таколи значення «i» зменшиться до 12 – зупинити його.

```
for (int i = 28; i >= 2; …)
{
    if (i == 12)
    {
        (…);
```

```
    }
}
Console.WriteLine(i);
```

Правильні відповіді:

1) i -= 2;
2) break.

9 крок. Натиснути кнопку *«Підтвердити відповідь»*. У разі якщо десь була допущена помилка з'явиться повідомлення про помилку,і поля, де вона була допущена будуть обведені червоним кольором, але якщо усі відповіді були коректні, то після підтвердження відповіді користувач зможе перейти до наступного завдання натиснувши на кнопку *«Наступне».*

10 крок. Користувач отримує наступне завдання.

Практичне завдання №5 з циклу for. Нижче наведено код. Скільки елементів «i» буде виведено на екран?

```
for (int i = 0; i <= 10; i++)
{
    Console.WriteLine(i);
}
```

Правильні відповіді:

1) 11;

11 крок. Натиснути кнопку *«Підтвердити відповідь»*. У разі якщо десь була допущена помилка з'явиться повідомлення про помилку,і поля, де вона була допущена будуть обведені червоним кольором, але якщо усі відповіді були коректні, то після підтвердження відповіді користувач зможе перейти до наступного завдання натиснувши на кнопку *«Наступне».*

12 крок. Користувач отримує наступне завдання.

Практичне завдання №6 з циклу for. Нижче наведено код. Перечисліть усі елементи «i» які будуть виведені на екран через кому або через пробіл.

```
for  (int i = 0; i <= 10; i++)
{
```

```
    if  (i == 6)
    {
        i = i + 5;
    }
 }
Console.WriteLine(i);
```

Правильні відповіді:

1) 01234511;
2) 0, 1, 2, 3, 4, 5, 11.

13 крок. Натиснути кнопку *«Підтвердити відповідь»*. У разі якщо десь була допущена помилка з’явиться повідомлення про помилку,і поля, де вона була допущена будуть обведені червоним кольором, але якщо усі відповіді були коректні, то після підтвердження відповіді користувач зможе перейти до наступного завдання натиснувши на кнопку *«Наступне».*

14 крок. Після того як користувач завершить всі завдання будь-якої практичної частини він отримає повідомлення про успішне завершення.

3.3. Блок-схема програми-симулятора

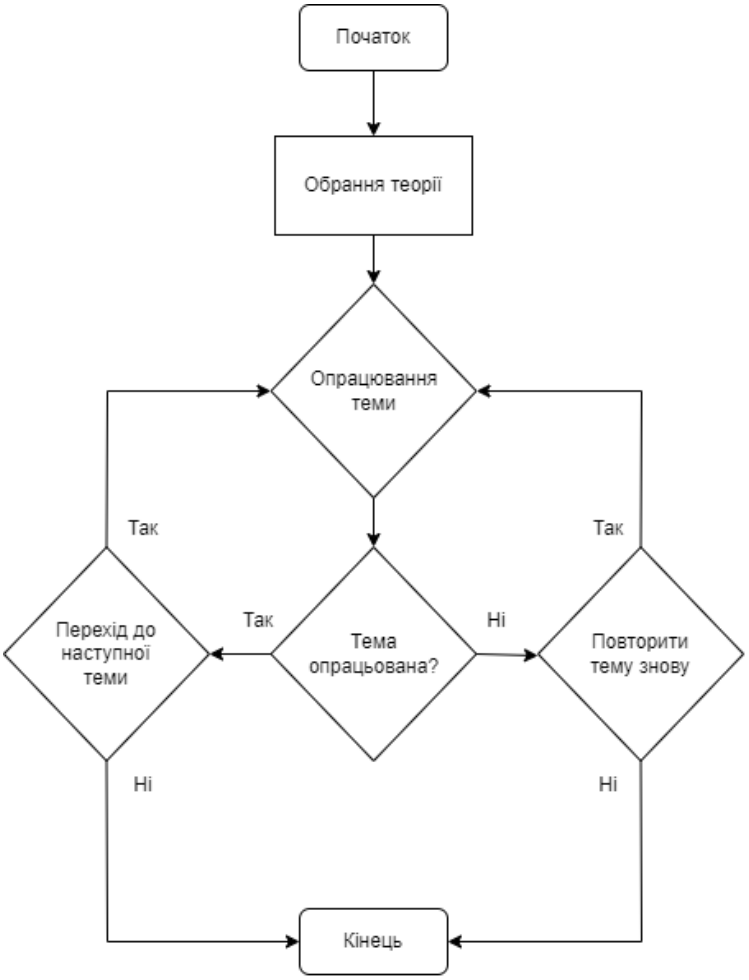

Рисунок 3.1 – Блок-схема роботи з теоретичним матеріалом

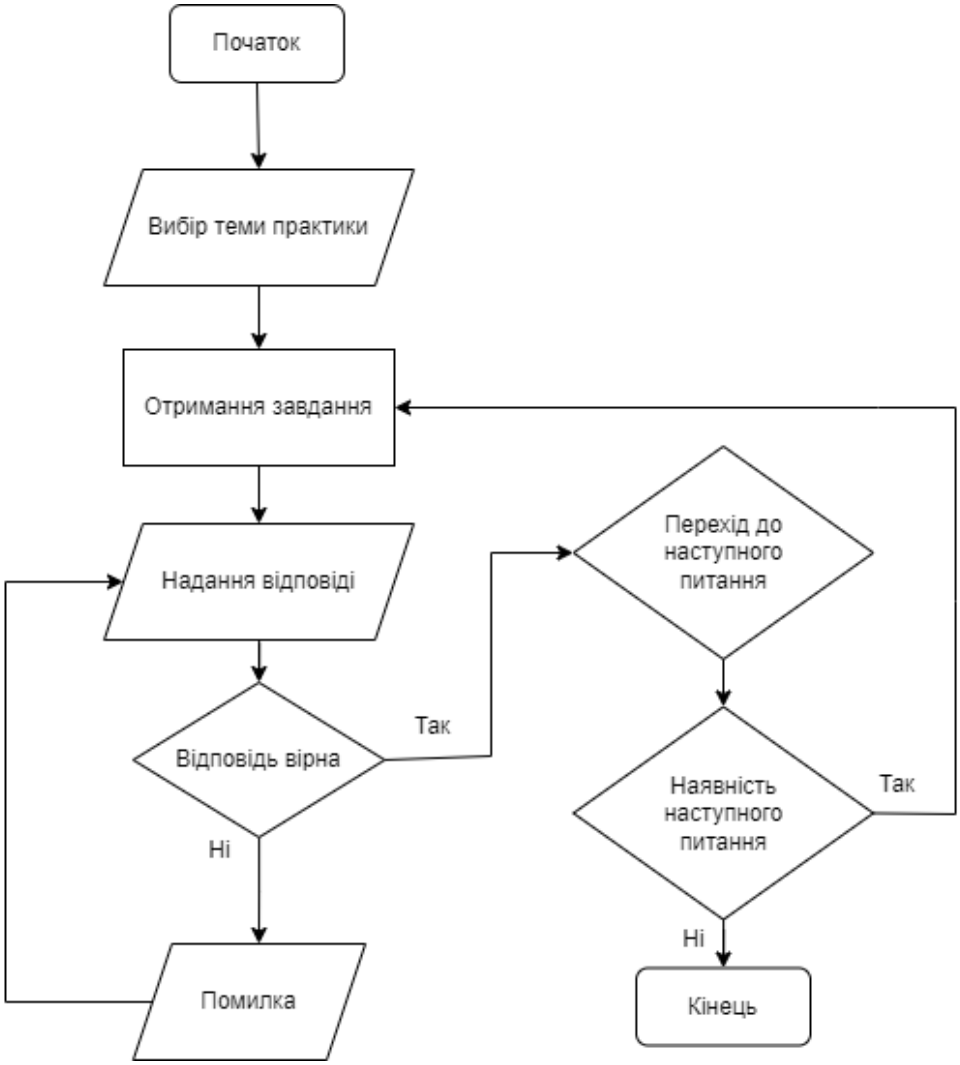

Рисунок 3.2 – Блок-схема роботи з практичним матеріалом

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис програмної реалізації

Для створення клієнтської частини додатку були використані такі технології як Windows Forms. Windows Forms впроваджує засоби для:

- створення структурованого інтерфейсу користувача;
- отримання інформації з форми;
- створення інтерактивних форм.

Спочатку потрібно створити стартовий екран користувача який буде містити в собі коротку інформацію про те, що саме робить даний застосунок (див. Рисунок 4.1).

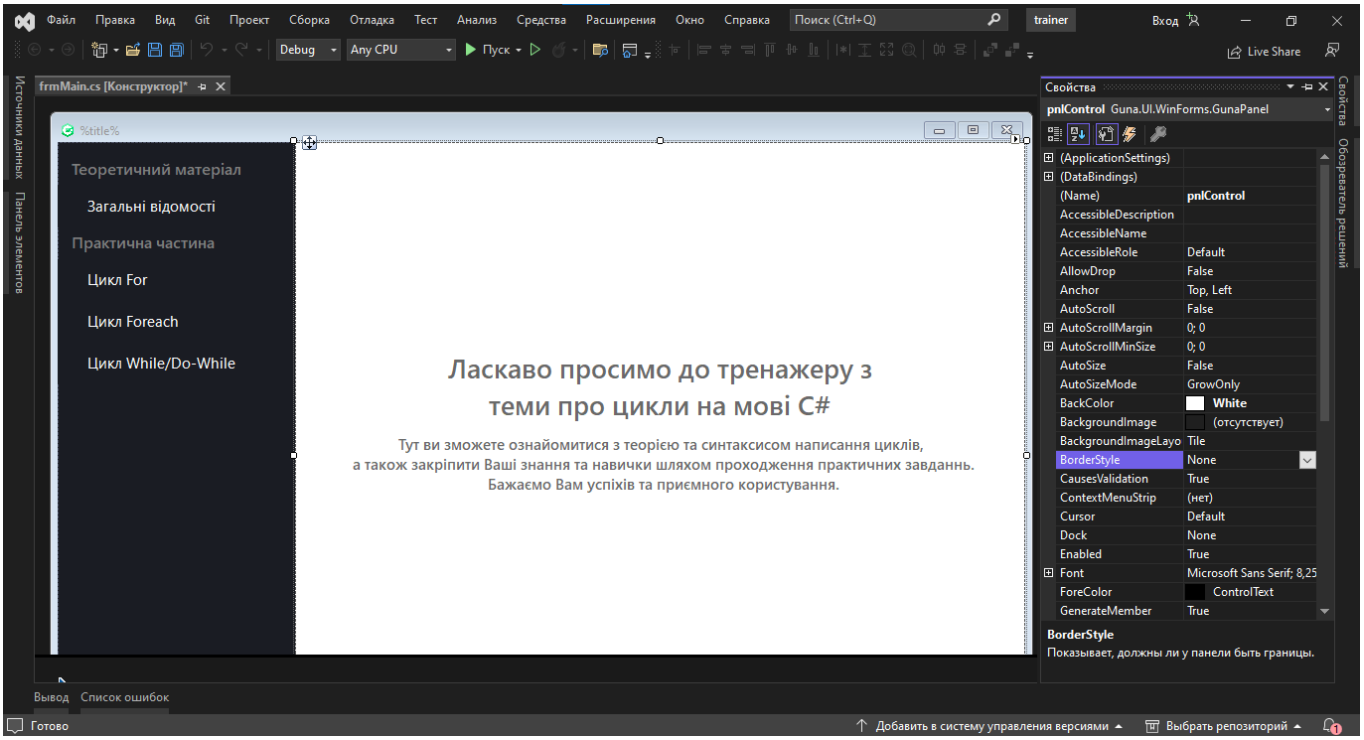

Рисунок 4.1 – Стартовий екран користувача

Після створення стартового екрану користувача розробляються основні частини застосунку такі як: теоретична частина та праткична (див. Рисунок 4.2 та 4.3).

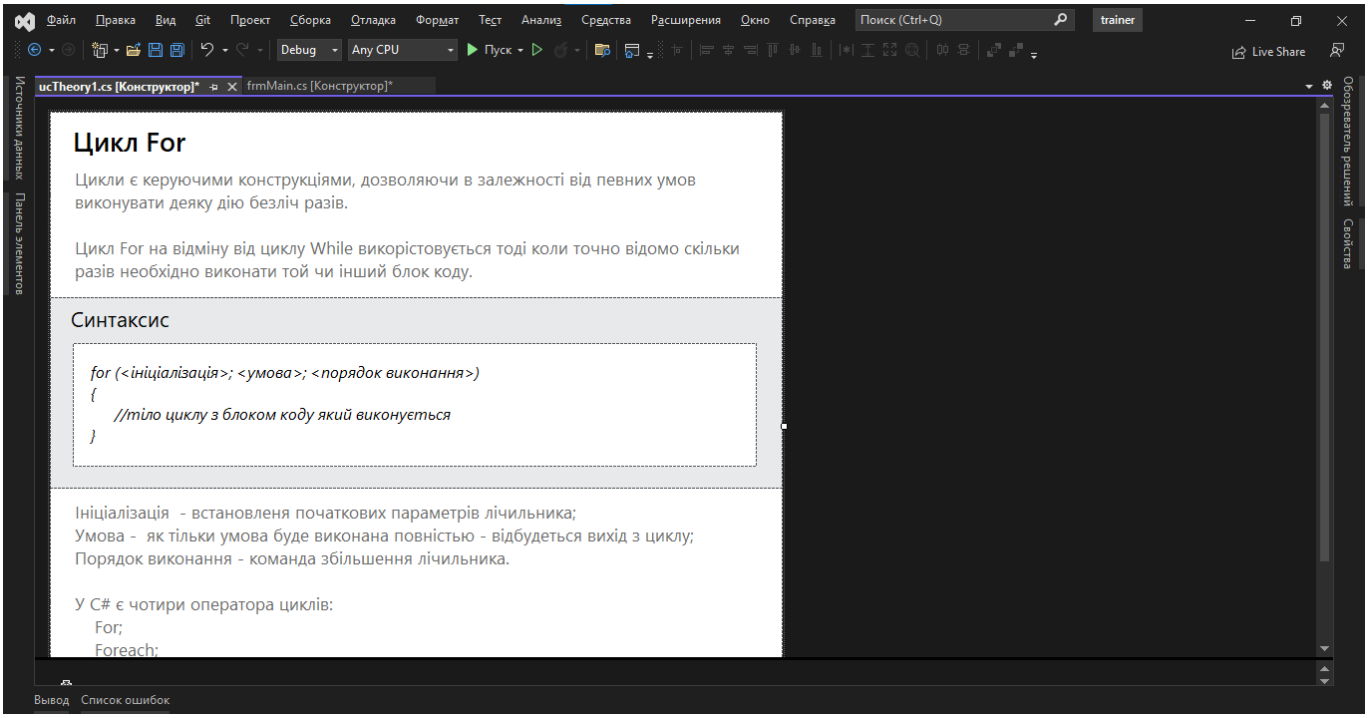

Рисунок 4.2 – Вікно теоретичної частини

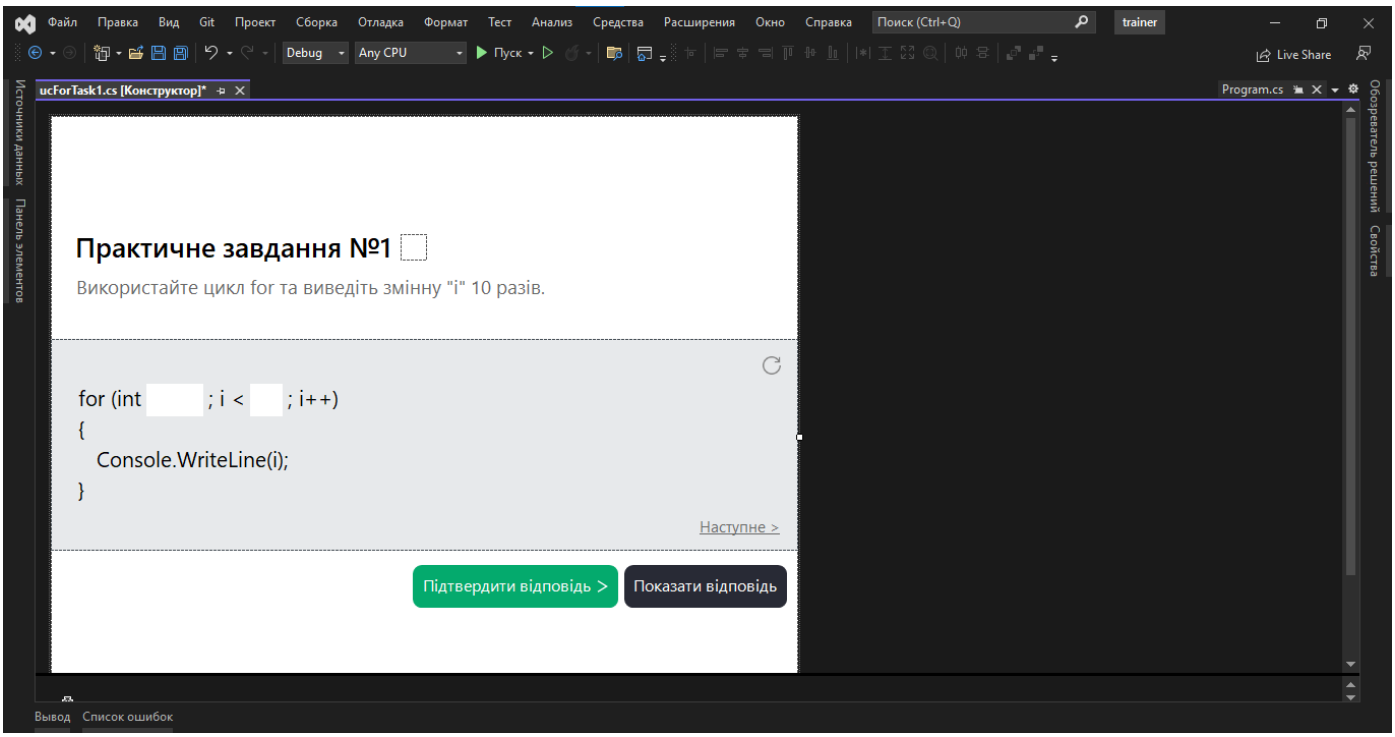

Рисунок – 4.3 – Вікно практичної частини

На даному етапі створюються вікна-слайди за допомогою яких користувач буде отримувати інформацію. Далі наведено невеликий код того як працюють вікна-слайди.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
    public partial class ucTheory1 : UserControl
    {
        private static ucTheory1 _instance;

        public static ucTheory1 instance
        {
            get
            {
                if (_instance == null)
                    _instance = new ucTheory1();
                return _instance;
            }
        }

        public ucTheory1()
        {
            InitializeComponent();

            ThemeChecker.Start();

            GraphicsHelper.DrawLineShadow(pnlFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);
            GraphicsHelper.DrawLineShadow(pnlCodeFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);
        }

        private void btnNext_Click(object sender, EventArgs e)
        {
            this.Controls.Clear();

            ucTheory2 theory = new ucTheory2();

            this.Controls.Add(theory);
        }

        private void ThemeChecker_Tick(object sender, EventArgs e)
        {
            if (frmMain.isNight) {
                lblTitle.ForeColor = Color.White;
```

```
            lblDesc1.ForeColor = Color.White;
            lblDesc2.ForeColor = Color.White;

            lblSyntax.ForeColor = Color.White;

            lblCodeSyntax.ForeColor = Color.White;
            pnlWhiteCodeFrame.BackColor = Color.FromArgb(40, 44, 52);
            pnlCodeFrame.BackColor = Color.FromArgb(17, 18, 23);

            lblPage.ForeColor = Color.White;

            pnlFrame.BackColor = Color.FromArgb(40, 44, 52);
        } else {
            lblTitle.ForeColor = Color.Black;

            lblDesc1.ForeColor = Color.DimGray;
            lblDesc2.ForeColor = Color.DimGray;

            lblSyntax.ForeColor = Color.Black;

            lblCodeSyntax.ForeColor = Color.Black;
            pnlWhiteCodeFrame.BackColor = Color.White;
            pnlCodeFrame.BackColor = Color.FromArgb(231, 233, 235);

            lblPage.ForeColor = Color.DimGray;

            pnlFrame.BackColor = Color.White;
        }
    }
  }
}
```

Кожне вікно в застосунку було створено за допомогою елемента *UserControl*. Цей елемент дозволяє створити окрему область де індивідуально для кожного вікна буде розташовуватись різного роду інформація та елементи керування, наприклад: текст, кнопки, блоки коду, підсказки та ін. Де *UcTheory/Uc*(*ForTask/ForEachTask/WhileDoWhile* – [номер]) – це всі елементи які будуть відображатися один за одним. Цей метод також застосовується при створенні практичної частини.

Кожне практичне завдання являє собою деякий блок коду у якого є пусті комірки (див. Рисунок 4.3) до яких потрібно вписати пропущений код. Нижче невеликий код як саме працюють вікна практичних частин.

```
private void btnSubmit_Click(object sender, EventArgs e)
    {
      if (!string.IsNullOrWhiteSpace(txtInput1.Text) || !string.IsNullOrWhiteSpace(txtInput2.Text)) {
        if (txtInput1.Text == "i = 0" && txtInput2.Text == "10") {
          pctBoxStatus.Visible = true;
          pctBoxStatus.Image = Properties.Resources.passed;

          lnklblNext.Visible = true;
        } else {
          pctBoxStatus.Visible = true;
          pctBoxStatus.Image = Properties.Resources.attention;

          lnklblNext.Visible = false;

          (new MsgBox(Type.WARNING, "Увага!", "Ви допустили помилку у коді. Будь-ласка перегляньте ще
     раз умову та код завдання.")).ShowDialog();
        }
      }
    }
```

Після того як користувач натисне кнопку *«Підтвердити відповідь»* спрацює не великий блок коду який звірить чи правильні відповіді надав користувач, у разі якщо відповідь була правильна зя’виться додаткова кнопка за допомогою якої можна перейти до наступного питання. Код який виконується при натискані кнопки наведено нижче.

```
private void lnklblNext_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
      this.Controls.Clear();

      ucForTask2 task = new ucForTask2();

      this.Controls.Add(task);
    }
```

В ньому замінюється один слайд на інший, які були заздалегідь створені та на яких індивідуально для кожного розміщені елементи керування.

У разі якщо користувач відповів не правильно блок коду у програмі викличе вікно з повідомлення про те, що користувач відповів не правильно та попросить його переглянути свою відповіді та надати іншу (див. Рисунок 4.3).

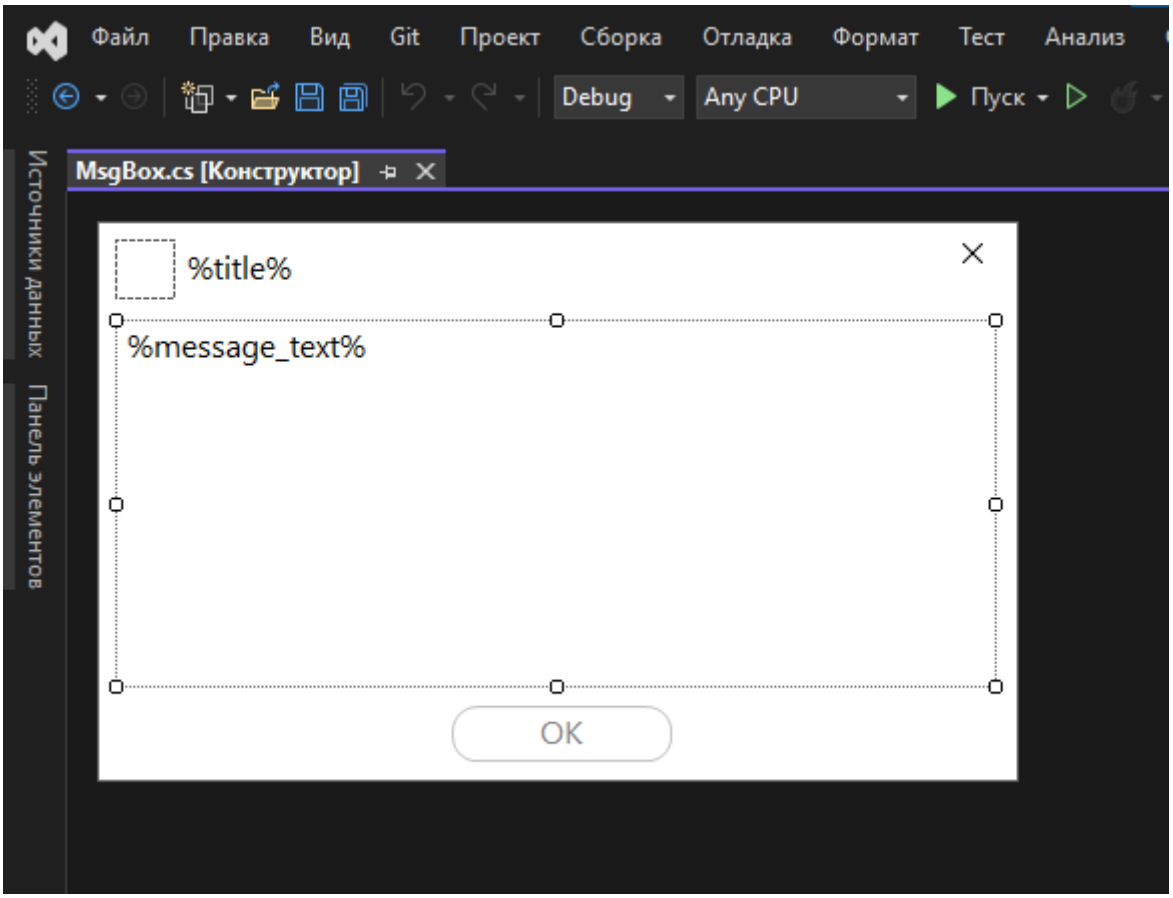

Рисунок 4.3 – Вікно повідомлення

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
    public enum Type { ERROR, WARNING, INFORMATION }

    public partial class MsgBox : Form
    {
        public MsgBox(Type type, string title, string message)
        {
            InitializeComponent();

            GraphicsHelper.ShadowForm(this);

            if (frmMain.isNight) {
                ControlButtonClose.IconColor = Color.White;
```

```
            lblTitle.ForeColor = Color.White;
            lblMessage.ForeColor = Color.White;

            btnOK.BaseColor = Color.FromArgb(40, 42, 53);
            btnOK.BorderColor = Color.Gray;

            pnlFrame.BackColor = Color.FromArgb(26, 28, 35);
        } else {
            ControlButtonClose.IconColor = Color.Black;

            lblTitle.ForeColor = Color.Black;
            lblMessage.ForeColor = Color.Black;

            btnOK.BaseColor = Color.White;
            btnOK.BorderColor = Color.Silver;

            pnlFrame.BackColor = Color.White;
        }

        if (type == Type.ERROR) {
            pctBoxWindowIcon.Image = Properties.Resources.error;

            btnOK.OnHoverBorderColor = Color.FromArgb(244, 67, 54);
            btnOK.OnHoverBaseColor = Color.FromArgb(244, 67, 54);
        }

        if (type == Type.WARNING) {
            pctBoxWindowIcon.Image = Properties.Resources.warning;

            btnOK.OnHoverBorderColor = Color.FromArgb(255, 202, 40);
            btnOK.OnHoverBaseColor = Color.FromArgb(255, 202, 40);
        }

        if (type == Type.INFORMATION) {
            pctBoxWindowIcon.Image = Properties.Resources.info;

            btnOK.OnHoverBorderColor = Color.FromArgb(33, 150, 243);
            btnOK.OnHoverBaseColor = Color.FromArgb(33, 150, 243);
        }

        lblTitle.Text = title;
        lblMessage.Text = message;

        if (lblMessage.Text.Length >= 260)
            this.Size = new Size(540, 350);
    }
```

```
        private void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Цей невеликий код зверху є прикладом того як працює показ повідомлень користувачу.

public enum Type { ERROR, WARNING, INFORMATION } – список станів які будуть містити в собі повідомлення всього їх 3. Для кожного з цих повідомлень заздалегідь були заготовлені стилі за допомогою яких при виклику їх можна було візуально відрізняти.

Де, *ERROR* – повідомлення про помилку;

Де, *WARNING* – повідомлення про попередження;

Де, *INFORMATION* – повідомлення про інформацію;

Для кожного з них є своє застосування, наприклад, коли користувач підтвердив неправильну відповідь із списку використається тип *WARNING* який попередить користувача про не коректність наданої відповіді.

```
lblTitle.Text = title;
lblMessage.Text = message;
```

За допомогою цих двох строчок коду розробник має можливість сам задати назву свого повідомлення і текст при виклику вікна, щоб не створювати багато окремих вікон. Це є універсальний спосіб сповіщення користувача у різних ситуаціях.

Створюваний додаток розрахований на отримання деяких знань з галузі мови програмування C# про цикли. Закріплення отриманих знань шляхом проходження практичних завдань, а також повинен бути інтуєтивно зрозумілий інтерфейс для користувача.

Створюваний додаток повинен мати такі вимоги:

- простий та інтуітивно зрозумілий інтерфейс;
- теоретичні відомості;
- практичні завдання.

Оскільки створюваний додаток не є масштабним, то для початку планується випустити версію з мінімальним базовим функціоналом та в подальшому вдосконалювати його.

Даний програмний продукт розроблювався за допомогою IDE MS Visual Studio. Тип роботи програмного забезпечення можна описати як інтерактивну презентацію в якій можливий вибір між теоретичними та практичними завданнями та зручним переходом між ними.

Перехід між сторінками реалізовано за допомогою кнопок *«Далі»* та *«Назад».* Робота кнопки *«Далі»* базується на закритті поточної сторінки та відкритті наступної. У разі закінчення роботи з завданнями теоретичного типу, необхідно обрати будь-яку з запропонованих практичних частин.

4.2. Інструкція по роботі з програмою

Після встановлення прогрманого продукту та його запуску у Вас з’явиться вікно з привітанням (див. Рисунок 4.1) та навігаційному меню яке знаходиться з ліва.

Навігаційне меню має такі пункти:

Теоретична частина:

- Загальні відомості.

Практична частина:

- Цикл For;
- Цикл Foreach;
- Цикл While/-Do-While.

Якщо користувач натискає «*Загальні відомості*» він починає ознайомлення з теоретичною частиною з теми про цикли. *Загальні відомості* містять в собі інформацію про 3 циклу, які присутні в симуляторі.

Після того як користувач ознайомився з теоретичним матеріалом, може приступити до виконання практичних завдань з починаючи з будь-якої теми.

По завершеню роботи користувач отримує необхідні базові навички по роботі з циклами. Які зможе потім застосувати у сфері з програмування на мові C#.

4.3. Тестування симулятору

Головне меню тренажера (див. Рисунок 4.2) дозволяє здійснити вибір між завданнями теоретичного та практичного вигляду. Перейти до теоретичного або практично матеріалу можна за допомогою бокового меню зліва. Переключатися між сторінками можна за допомогою кнопок *«Далі»* та *«Назад»*. Переключатися між практичними завданнями можна за допомогою кнопок *«Наступне»* та *«Назад»*.

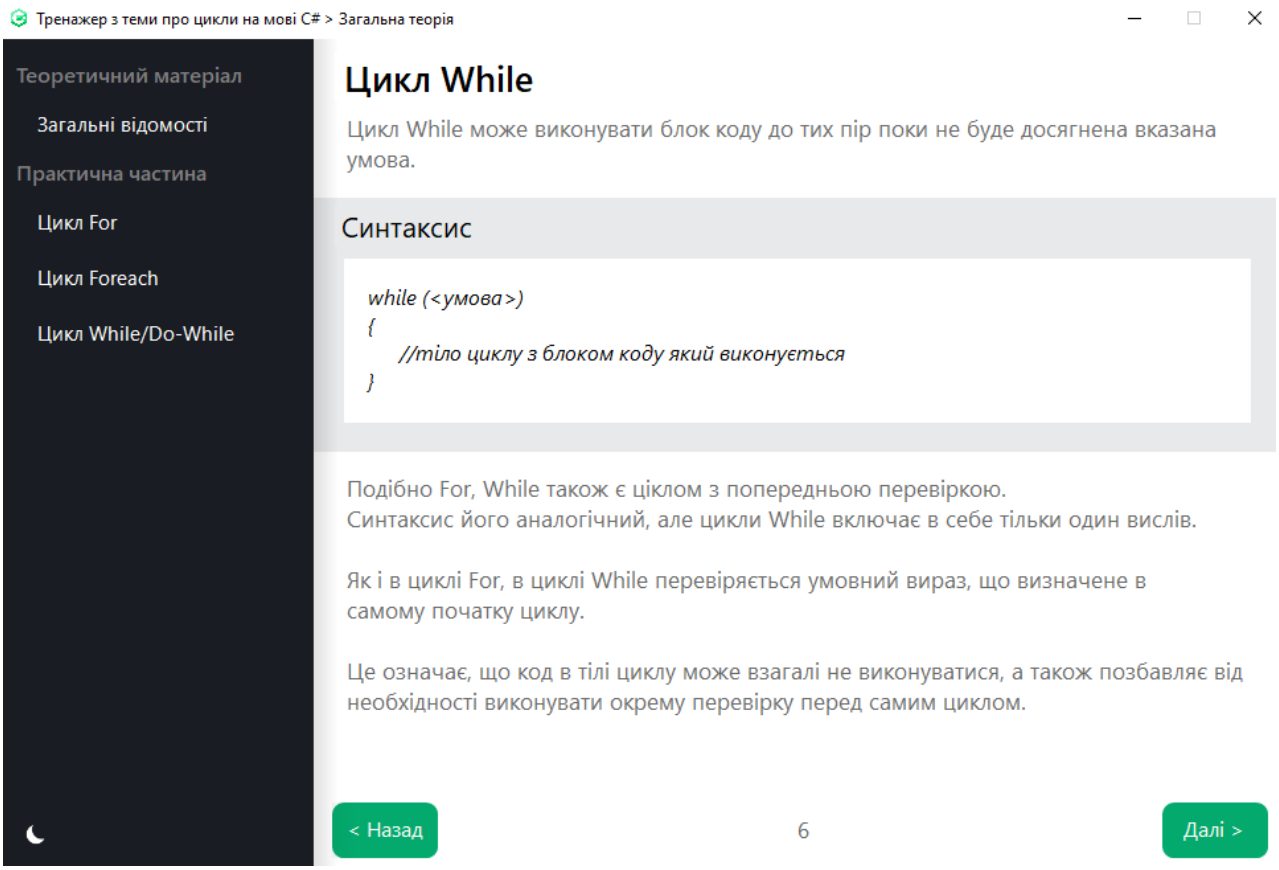

Рисунок 4.4 – Головний екран симулятора

Після опанування теоретичної частини користувач має змогу обрати одну з практичних частин або відразу приступити до їх виконання (див. Рисунок 4.3).

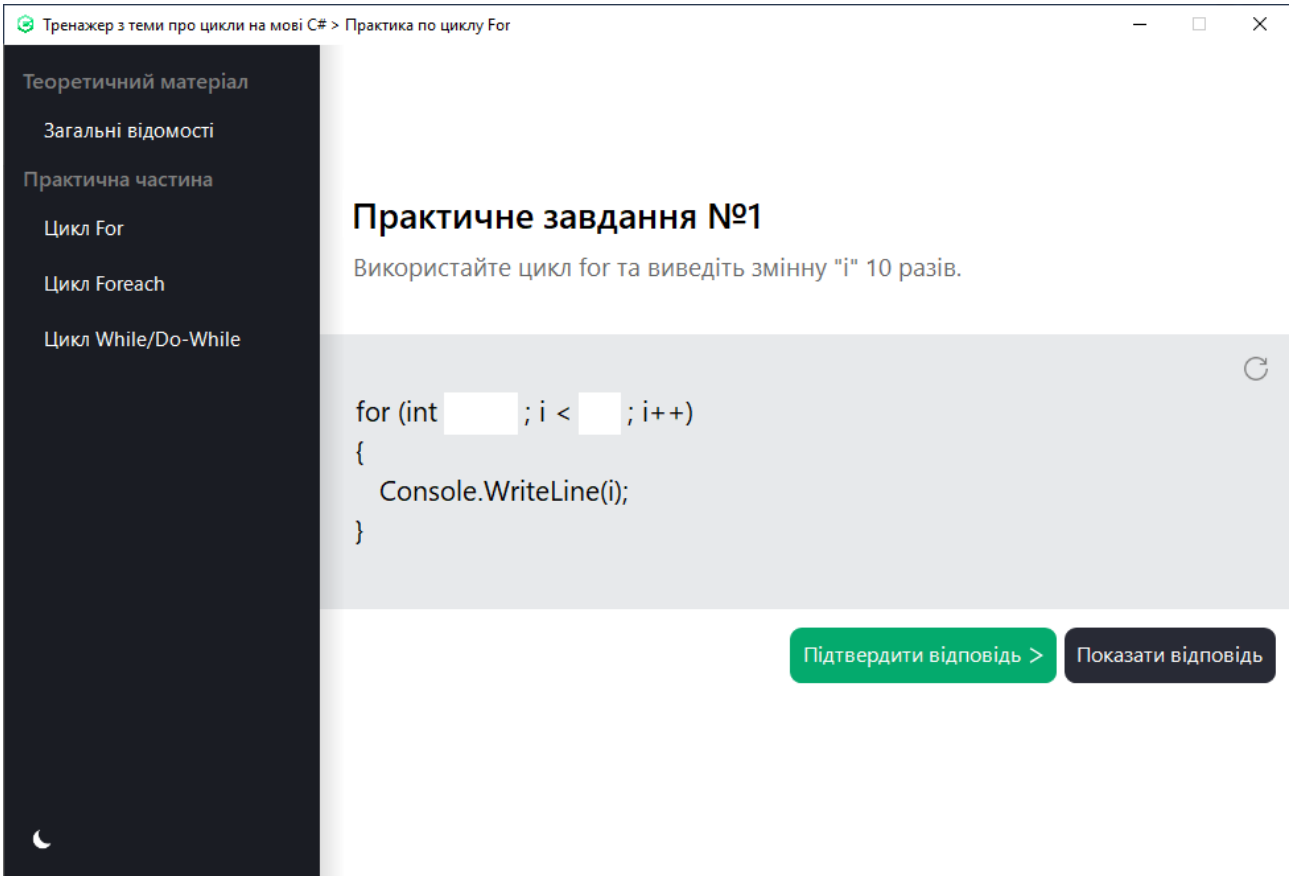

Рисунок 4.5 – Практична частина

При роботи з практичною частиною симулятора користувач отримує завдання (деякий код) та поле для вводу для відповіді (див. Рисунок 4.4). Після вводу певної відповіді та натиснення на кнопку «*Підтвердити відповідь*» користувач отримує повідомлення про коректність наданої відповіді, але якщо відповідь була неправильна (див. Рисунок 4.5) отримає вікно з попередженням, а також завжди має доступ до кнопки «*Підказка*» за допомогою якої має змогу переглянути правильну відповіді на запитання (див. Рисунок 4.6).

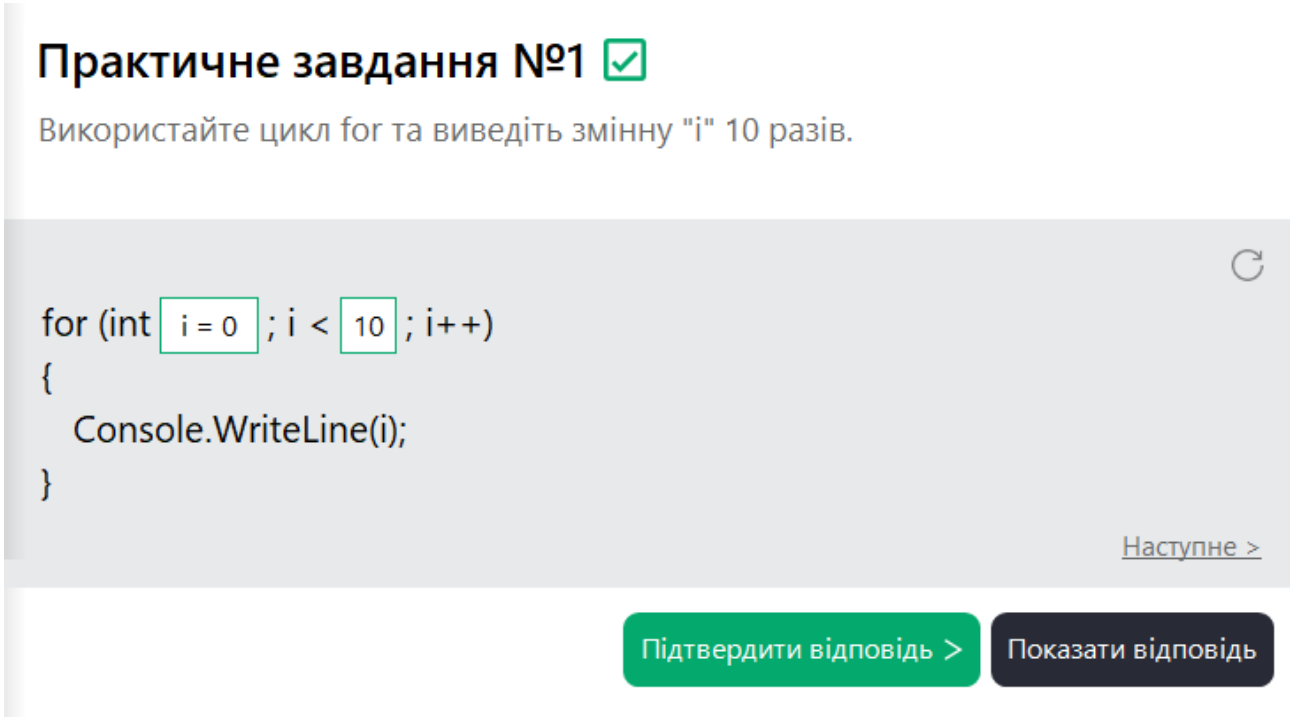

Рисунок 4.6 – Вікно після надання правильної відповіді

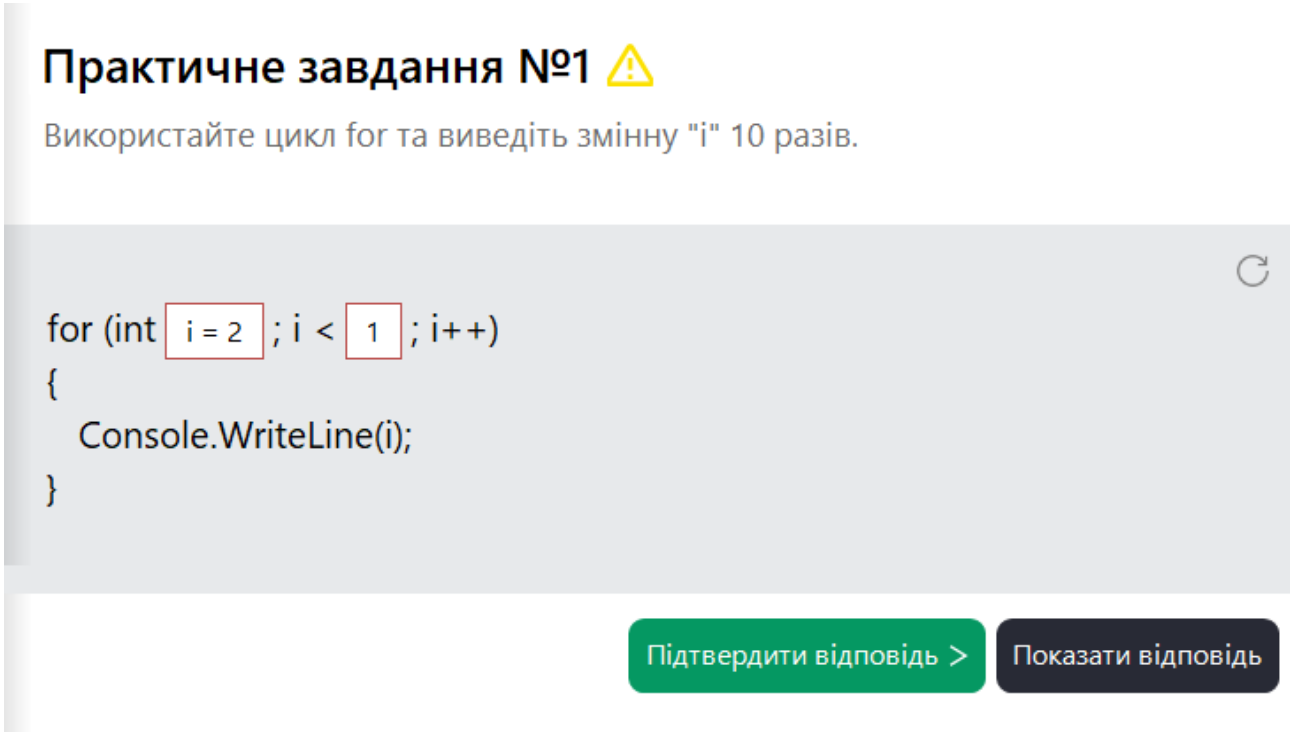

Рисунок 4.7 – Вікно після надання неправильної відповіді

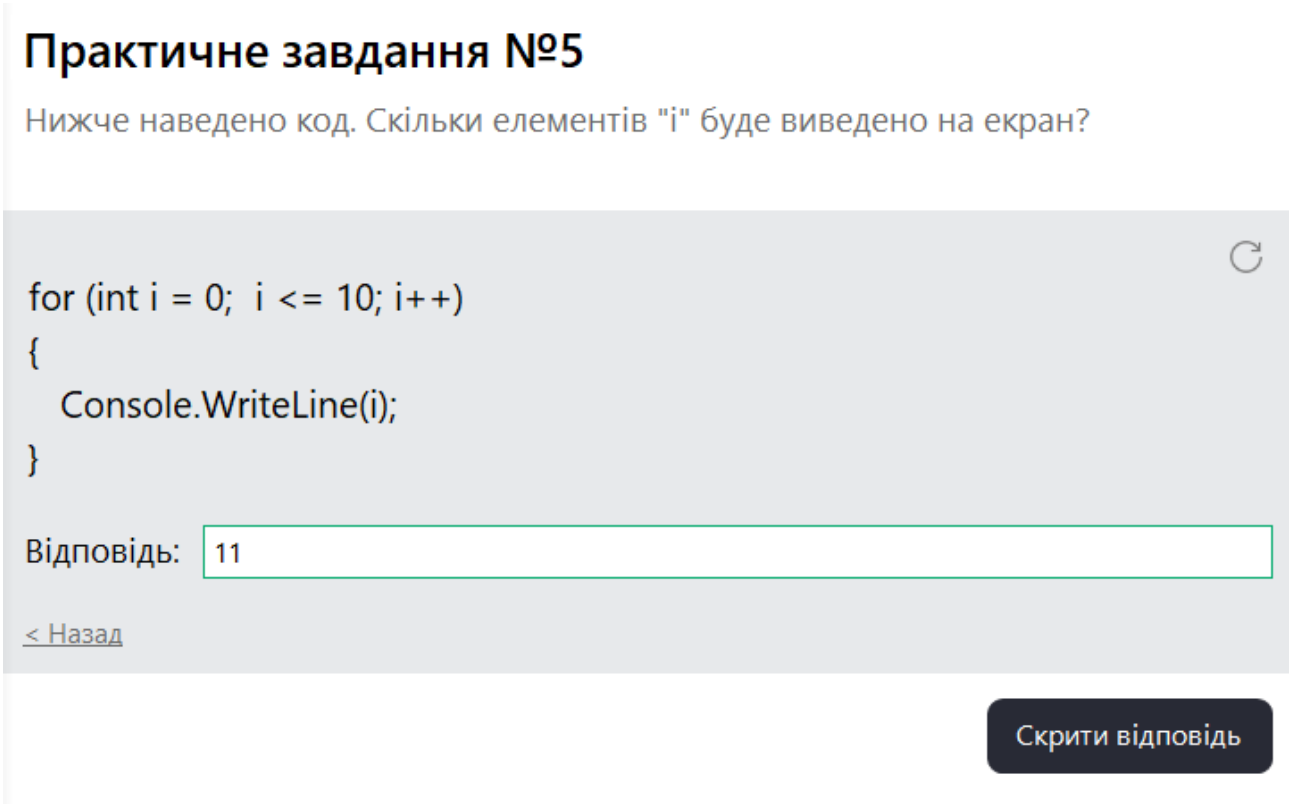

Рисунок 4.8 – Кнопка «Показати відповідь»

Для доступу до практичних завдань під час роботи з тренажером достатньо натиснути кнопку «*Загальні відомості*» з бокового меню та знайти потрібний матеріал який необхідний для виконання завдання.

З будь-якого практичного завдання є можливість перейти до виконання інших практичних завдань, або повернутись до практичної частини.

Даний симулятор за допомоги платформи .Net Framework поводить себе однаково та працює стабільно на операційних системах Windows.

ВИСНОВКИ

Незважаючи на те, що технологічна основа навчального процесу у вищий школі, у тому числі і сучасні інформаційні технології, швидко розвиваються, застосування сучасних комп’ютерних технологій в навчальному процесі не тільки створює умови для більш ефективної самостійної роботи студентів, сприяє індивідуалізації процесу підготовки фахівців, а і суттєво змінює форми і зміст комунікацій між викладачем і студентом. За допомогою комп’ютерних технологій, незважаючи на незмінні тенденції до зменшення аудиторних годин, прямий і зворотній зв’язок «викладач-студент» стає більш інтенсивним і активним.

Метою проекту є «Алгоритмізація та розробка програмного забезпечення з теми «Цикли мови C#» , яка повністю реалізована.

В проекті описано такі пункти:

- Постановка задачі;
- Типи комп'ютерних тренажерів;
- Середовища та мови програмування, середовища для розробки програмного забезпечення;
- Практична частина;
- Огляд матеріалу з теми;
- Алгоритм робити тренажера.

Для програмної реалізації використані такі засоби:

- MS Visual Studio;
- Windows Froms;
- .Net Framework.

Розроблений програмний продукт може бути використаний як інструмент контролю та самоконтроля засвоєння знань з теми.

Створений програмний продукт апробовано у вигляді тез на конференціях.

Незважаючи на те, що технологічна основа навчального процесу у вищий школі, у тому числі і сучасні інформаційні технології, швидко розвиваються, застосування сучасних комп’ютерних технологій в навчальному процесі не

тільки створює умови для більш ефективної самостійної роботи студентів, сприяє індивідуалізації процесу підготовки фахівців, а і суттєво змінює форми і зміст комунікацій між викладачем і студентом. За допомогою комп’ютерних технологій, незважаючи на незмінні тенденції до зменшення аудиторних годин, прямий і зворотній зв'язок «викладач-студент» стає більш інтенсивним і активним.

Отже, використання програм-симуляторів як компонент навчального середовища створює можливості до ефективної інтеграції самостійної роботи в навчальний процес та без залучення додаткових витрат аудиторного часу прибрати прогалини в знаннях здобувачів освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ольховська О. В. Методичні рекомендації до виконання кваліфікаційної роботи / О. В. Ольховська, О. О. Черненко., С. В. Гаркуша. // ПУЕТ. – 2022. (дата звернення: 09.12.2023)

2. C Sharp Programming Language [Електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/C_Sharp. (дата звернення: 17.10.2023)

3. Інтегроване середовище розробки [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. – Режим доступу:

https://uk.wikipedia.org/wiki/Інтегроване_середовище_розробки. (дата звернення: 12.11.2023)

4. Visual Studio [Електронний ресурс]. – Режим доступу: URL: https://visualstudio.microsoft.com/vs/. (дата звернення: 14.11.2023)

5. Windows Forms [Електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/Windows_Forms. (дата звернення: 19.11.2023)

6. .NET Platform [Електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/.NET_Framework. (дата звернення: 25.11.2023)

7. CLR. Microsoft Docs [Електронний ресурс]. – Режим доступу: URL: https://docs.microsoft.com/en-us/dotnet/standard/clr. (дата звернення: 29.11.2023)

8. User Interface. Wikipedia [Електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/User_interface (дата звернення: 11.11.2023)

9. User experience. Wikipedia [Електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/User_experience (дата звернення: 11.11.2023)

10. Stack Overflow. [Електронний ресурс]. – Режим доступу: URL: https://stackoverflow.com/ (дата звернення: 11.11.2023)

11. W3Schools. [Електронний ресурс]. – Режим доступу: URL: https://www.w3schools.com/ (дата звернення: 11.11.2023)

12. Github. [Електронний ресурс]. – Режим доступу: URL: https://github.com/ (дата звернення: 11.11.2023)

ДОДАТОК А

ГОЛОВНЕ ВІКНО

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
   public partial class frmMain : Form
   {
      public static bool isNight = false;

      public frmMain()
      {
         InitializeComponent();

         this.Text = "Тренажер з теми про цикли на мові C#";

         pnlControl.BackColor = Color.White;

         GraphicsHelper.DrawLineShadow(pnlControl, Color.Black, 30, 18,
          Guna.UI.WinForms.VerHorAlign.VerticalLeft);
      }

      //THEORY

      private void btnTheory_Click(object sender, EventArgs e)
      {
         this.Text = "Тренажер з теми про цикли на мові C# > Загальна теорія";

         if (!pnlControl.Controls.Contains(ucTheory1.instance)) {
            pnlControl.Controls.Add(ucTheory1.instance);
            ucTheory1.instance.Dock = DockStyle.Fill;
            ucTheory1.instance.BringToFront();
         } else {
            ucTheory1.instance.BringToFront();
         }
      }

      //PRACTICE

      private void btnPracticeFor_Click(object sender, EventArgs e)
```

```
{
    this.Text = "Тренажер з теми про цикли на мові C# > Практика по циклу For";

    if (!pnlControl.Controls.Contains(ucForTask1.instance)) {
        pnlControl.Controls.Add(ucForTask1.instance);
        ucForTask1.instance.Dock = DockStyle.Fill;
        ucForTask1.instance.BringToFront();
    } else {
        ucForTask1.instance.BringToFront();
    }
}

private void btnPracticeForeach_Click(object sender, EventArgs e)
{
    this.Text = "Тренажер з теми про цикли на мові C# > Практика по циклу Foreach";

    if (!pnlControl.Controls.Contains(ucForeachTask1.instance)) {
        pnlControl.Controls.Add(ucForeachTask1.instance);
        ucForeachTask1.instance.Dock = DockStyle.Fill;
        ucForeachTask1.instance.BringToFront();
    } else {
        ucForeachTask1.instance.BringToFront();
    }
}

private void btnPracticeForWhileDoWhile_Click(object sender, EventArgs e)
{
    this.Text = "Тренажер з теми про цикли на мові C# > Практика по циклу While/Do-While";

    if (!pnlControl.Controls.Contains(ucWhileDoWhileTask1.instance)) {
        pnlControl.Controls.Add(ucWhileDoWhileTask1.instance);
        ucWhileDoWhileTask1.instance.Dock = DockStyle.Fill;
        ucWhileDoWhileTask1.instance.BringToFront();
    } else {
        ucWhileDoWhileTask1.instance.BringToFront();
    }
}

//OTHER

private void btnChangeTheme_Click(object sender, EventArgs e)
{
    if (!isNight) {
        isNight = true;
```

```
            Tips.SetToolTip(btnChangeTheme, "Увімкнути світлу тему");

            btnChangeTheme.Image = Properties.Resources.sun;
            btnChangeTheme.ImageOffsetX = -1;

            lblGreetings.ForeColor = Color.White;
            lblDesc.ForeColor = Color.White;

            pnlControl.BackColor = Color.FromArgb(40, 44, 52);
        } else {
            isNight = false;

            Tips.SetToolTip(btnChangeTheme, "Увімкнути нічну тему");

            btnChangeTheme.Image = Properties.Resources.moon;

            lblGreetings.ForeColor = Color.DimGray;
            lblDesc.ForeColor = Color.DimGray;

            pnlControl.BackColor = Color.White;
        }
    }

    private void frmMain_FormClosing(object sender, FormClosingEventArgs e)
    {
        this.Dispose();

        Environment.Exit(1);
    }
  }
}
```

ВІКНО ПОВІДОМЛЕННЯ

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
   public enum Type { ERROR, WARNING, INFORMATION }

   public partial class MsgBox : Form
   {
      public MsgBox(Type type, string title, string message)
      {
         InitializeComponent();

         GraphicsHelper.ShadowForm(this);

         if (frmMain.isNight) {
            ControlButtonClose.IconColor = Color.White;

            lblTitle.ForeColor = Color.White;
            lblMessage.ForeColor = Color.White;

            btnOK.BaseColor = Color.FromArgb(40, 42, 53);
            btnOK.BorderColor = Color.Gray;

            pnlFrame.BackColor = Color.FromArgb(26, 28, 35);
         } else {
            ControlButtonClose.IconColor = Color.Black;

            lblTitle.ForeColor = Color.Black;
            lblMessage.ForeColor = Color.Black;

            btnOK.BaseColor = Color.White;
            btnOK.BorderColor = Color.Silver;

            pnlFrame.BackColor = Color.White;
         }

         if (type == Type.ERROR) {
            pctBoxWindowIcon.Image = Properties.Resources.error;

            btnOK.OnHoverBorderColor = Color.FromArgb(244, 67, 54);
            btnOK.OnHoverBaseColor = Color.FromArgb(244, 67, 54);
         }
```

```
            if (type == Type.WARNING) {
                pctBoxWindowIcon.Image = Properties.Resources.warning;

                btnOK.OnHoverBorderColor = Color.FromArgb(255, 202, 40);
                btnOK.OnHoverBaseColor = Color.FromArgb(255, 202, 40);
            }

            if (type == Type.INFORMATION) {
                pctBoxWindowIcon.Image = Properties.Resources.info;

                btnOK.OnHoverBorderColor = Color.FromArgb(33, 150, 243);
                btnOK.OnHoverBaseColor = Color.FromArgb(33, 150, 243);
            }

            lblTitle.Text = title;
            lblMessage.Text = message;

            if (lblMessage.Text.Length >= 260)
                this.Size = new Size(540, 350);
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

ВІКНО З ТЕОРЕТИЧНИМ МАТЕРІАЛОМ

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
    public partial class ucTheory1 : UserControl
    {
        private static ucTheory1 _instance;

        public static ucTheory1 instance
        {
            get
            {
                if (_instance == null)
                    _instance = new ucTheory1();
                return _instance;
            }
        }

        public ucTheory1()
        {
            InitializeComponent();

            ThemeChecker.Start();

            GraphicsHelper.DrawLineShadow(pnlFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);
            GraphicsHelper.DrawLineShadow(pnlCodeFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);
        }

        private void btnNext_Click(object sender, EventArgs e)
        {
            this.Controls.Clear();

            ucTheory2 theory = new ucTheory2();

            this.Controls.Add(theory);
        }

        private void ThemeChecker_Tick(object sender, EventArgs e)
        {
            if (frmMain.isNight) {
```

```
            lblTitle.ForeColor = Color.White;

            lblDesc1.ForeColor = Color.White;
            lblDesc2.ForeColor = Color.White;

            lblSyntax.ForeColor = Color.White;

            lblCodeSyntax.ForeColor = Color.White;
            pnlWhiteCodeFrame.BackColor = Color.FromArgb(40, 44, 52);
            pnlCodeFrame.BackColor = Color.FromArgb(17, 18, 23);

            lblPage.ForeColor = Color.White;

            pnlFrame.BackColor = Color.FromArgb(40, 44, 52);
        } else {
            lblTitle.ForeColor = Color.Black;

            lblDesc1.ForeColor = Color.DimGray;
            lblDesc2.ForeColor = Color.DimGray;

            lblSyntax.ForeColor = Color.Black;

            lblCodeSyntax.ForeColor = Color.Black;
            pnlWhiteCodeFrame.BackColor = Color.White;
            pnlCodeFrame.BackColor = Color.FromArgb(231, 233, 235);

            lblPage.ForeColor = Color.DimGray;

            pnlFrame.BackColor = Color.White;
        }
    }
  }
}
```

ВІКНО З ПРАКТИЧНИМ МАТЕРІАЛОМ

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
   public partial class ucForeachTask1 : UserControl
   {
      private static ucForeachTask1 _instance;

      public static ucForeachTask1 instance
      {
         get
         {
            if (_instance == null)
               _instance = new ucForeachTask1();
            return _instance;
         }
      }

      public ucForeachTask1()
      {
         InitializeComponent();

         ThemeChecker.Start();

         GraphicsHelper.DrawLineShadow(pnlFrame, Color.Black, 30, 18,
          Guna.UI.WinForms.VerHorAlign.VerticalLeft);
         GraphicsHelper.DrawLineShadow(pnlCodeFrame, Color.Black, 30, 18,
          Guna.UI.WinForms.VerHorAlign.VerticalLeft);

         lnklblNext.Visible = false;

         pctBoxStatus.Visible = false;
      }

      private void btnRefresh_Click(object sender, EventArgs e)
      {
         pctBoxStatus.Visible = false;

         txtInput1.Text = "";
         txtInput2.Text = "";
         txtInput3.Text = "";
```

```
    txtInput1.BorderColor = Color.White;
    txtInput2.BorderColor = Color.White;
    txtInput3.BorderColor = Color.White;

    lnklblNext.Visible = false;

    btnSubmit.Visible = true;

    btnShowAnswer.Text = "Показати відповідь";
}

private void lnklblNext_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.Controls.Clear();

    ucForeachTask2 task = new ucForeachTask2();

    this.Controls.Add(task);
}

private void btnSubmit_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(txtInput1.Text) || !string.IsNullOrWhiteSpace(txtInput2.Text) ||
     !string.IsNullOrWhiteSpace(txtInput3.Text)) {
        if (txtInput1.Text == "string" && txtInput2.Text == "names" && txtInput3.Text == "item") {
            pctBoxStatus.Visible = true;
            pctBoxStatus.Image = Properties.Resources.passed;

            lnklblNext.Visible = true;
        } else {
            pctBoxStatus.Visible = true;
            pctBoxStatus.Image = Properties.Resources.attention;

            lnklblNext.Visible = false;

            (new MsgBox(Type.WARNING, "Увага!", "Ви допустили помилку у коді. Будь-ласка перегляньте ще
     раз умову та код завдання.")).ShowDialog();
        }
    }
}

private void btnShowAnswer_Click(object sender, EventArgs e)
{
    if (btnShowAnswer.Text == "Показати відповідь") {
        txtInput1.Text = "string";
        txtInput2.Text = "names";
        txtInput3.Text = "item";
```

```
            btnSubmit.Visible = false;

            btnShowAnswer.Text = "Скрити відповідь";
        } else if (btnShowAnswer.Text == "Скрити відповідь") {
            txtInput1.Text = "";
            txtInput2.Text = "";
            txtInput3.Text = "";

            btnSubmit.Visible = true;

            btnShowAnswer.Text = "Показати відповідь";
        }
    }

    private void ThemeChecker_Tick(object sender, EventArgs e)
    {
        if (frmMain.isNight) {
            lblPracticeTask.ForeColor = Color.White;
            lblTask.ForeColor = Color.White;
            lblCode.ForeColor = Color.White;

            txtInput1.ForeColor = Color.White;
            txtInput2.ForeColor = Color.White;
            txtInput3.ForeColor = Color.White;
            txtInput1.FocusedForeColor = Color.White;
            txtInput2.FocusedForeColor = Color.White;
            txtInput3.FocusedForeColor = Color.White;

            txtInput1.BaseColor = Color.FromArgb(40, 44, 52);
            txtInput2.BaseColor = Color.FromArgb(40, 44, 52);
            txtInput3.BaseColor = Color.FromArgb(40, 44, 52);
            txtInput1.FocusedBaseColor = Color.FromArgb(40, 44, 52);
            txtInput2.FocusedBaseColor = Color.FromArgb(40, 44, 52);
            txtInput3.FocusedBaseColor = Color.FromArgb(40, 44, 52);

            txtInput1.BorderColor = Color.FromArgb(40, 44, 52);
            txtInput2.BorderColor = Color.FromArgb(40, 44, 52);
            txtInput3.BorderColor = Color.FromArgb(40, 44, 52);
            txtInput1.FocusedBorderColor = Color.FromArgb(40, 44, 52);
            txtInput2.FocusedBorderColor = Color.FromArgb(40, 44, 52);
            txtInput3.FocusedBorderColor = Color.FromArgb(40, 44, 52);

            pnlCodeFrame.BackColor = Color.FromArgb(17, 18, 23);
            pnlFrame.BackColor = Color.FromArgb(40, 44, 52);

            btnShowAnswer.BaseColor = Color.FromArgb(17, 18, 23);
```

```
} else {
    lblPracticeTask.ForeColor = Color.Black;
    lblTask.ForeColor = Color.DimGray;
    lblCode.ForeColor = Color.Black;

    txtInput1.ForeColor = Color.Black;
    txtInput2.ForeColor = Color.Black;
    txtInput3.ForeColor = Color.Black;
    txtInput1.FocusedForeColor = Color.Black;
    txtInput2.FocusedForeColor = Color.Black;
    txtInput3.FocusedForeColor = Color.Black;

    txtInput1.BaseColor = Color.White;
    txtInput2.BaseColor = Color.White;
    txtInput3.BaseColor = Color.White;
    txtInput1.FocusedBaseColor = Color.White;
    txtInput2.FocusedBaseColor = Color.White;
    txtInput3.FocusedBaseColor = Color.White;

    txtInput1.BorderColor = Color.White;
    txtInput2.BorderColor = Color.White;
    txtInput3.BorderColor = Color.White;
    txtInput1.FocusedBorderColor = Color.White;
    txtInput2.FocusedBorderColor = Color.White;
    txtInput3.FocusedBorderColor = Color.White;

    pnlCodeFrame.BackColor = Color.FromArgb(231, 233, 235);
    pnlFrame.BackColor = Color.White;

    btnShowAnswer.BaseColor = Color.FromArgb(40, 42, 53);
}

if (!string.IsNullOrWhiteSpace(txtInput1.Text) || !string.IsNullOrWhiteSpace(txtInput2.Text) ||
 !string.IsNullOrWhiteSpace(txtInput3.Text)) {
    if (txtInput1.Text == "string")
        txtInput1.BorderColor = Color.FromArgb(4, 170, 109);
    else
        txtInput1.BorderColor = Color.FromArgb(185, 74, 72);

    if (txtInput2.Text == "names")
        txtInput2.BorderColor = Color.FromArgb(4, 170, 109);
    else
        txtInput2.BorderColor = Color.FromArgb(185, 74, 72);

    if (txtInput3.Text == "item")
        txtInput3.BorderColor = Color.FromArgb(4, 170, 109);
    else
```

```
                txtInput3.BorderColor = Color.FromArgb(185, 74, 72);
            }
        }
    }
}
```

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
    public partial class ucForTask1 : UserControl
    {
        private static ucForTask1 _instance;

        public static ucForTask1 instance
        {
            get
            {
                if (_instance == null)
                    _instance = new ucForTask1();
                return _instance;
            }
        }

        public ucForTask1()
        {
            InitializeComponent();

            ThemeChecker.Start();

            GraphicsHelper.DrawLineShadow(pnlFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);
            GraphicsHelper.DrawLineShadow(pnlCodeFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);

            lnklblNext.Visible = false;

            pctBoxStatus.Visible = false;
        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {
            pctBoxStatus.Visible = false;

            txtInput1.Text = "";
            txtInput2.Text = "";

            txtInput1.BorderColor = Color.White;
            txtInput2.BorderColor = Color.White;
```

```
    lnklblNext.Visible = false;

    btnSubmit.Visible = true;

    btnShowAnswer.Text = "Показати відповідь";
}

private void lnklblNext_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.Controls.Clear();

    ucForTask2 task = new ucForTask2();

    this.Controls.Add(task);
}

private void btnSubmit_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(txtInput1.Text) || !string.IsNullOrWhiteSpace(txtInput2.Text)) {
        if (txtInput1.Text == "i = 0" && txtInput2.Text == "10") {
            pctBoxStatus.Visible = true;
            pctBoxStatus.Image = Properties.Resources.passed;

            lnklblNext.Visible = true;
        } else {
            pctBoxStatus.Visible = true;
            pctBoxStatus.Image = Properties.Resources.attention;

            lnklblNext.Visible = false;

            (new MsgBox(Type.WARNING, "Увага!", "Ви допустили помилку у коді. Будь-ласка перегляньте ще
     раз умову та код завдання.")).ShowDialog();
        }
    }
}

private void btnShowAnswer_Click(object sender, EventArgs e)
{
    if (btnShowAnswer.Text == "Показати відповідь") {
        txtInput1.Text = "i = 0";
        txtInput2.Text = "10";

        btnSubmit.Visible = false;

        btnShowAnswer.Text = "Скрити відповідь";
    } else if (btnShowAnswer.Text == "Скрити відповідь") {
```

```
        txtInput1.Text = "";
        txtInput2.Text = "";

        btnSubmit.Visible = true;

        btnShowAnswer.Text = "Показати відповідь";
    }
}

private void ThemeChecker_Tick(object sender, EventArgs e)
{
    if (frmMain.isNight) {
        lblPracticeTask.ForeColor = Color.White;
        lblTask.ForeColor = Color.White;
        lblCode.ForeColor = Color.White;

        txtInput1.ForeColor = Color.White;
        txtInput2.ForeColor = Color.White;
        txtInput1.FocusedForeColor = Color.White;
        txtInput2.FocusedForeColor = Color.White;

        txtInput1.BaseColor = Color.FromArgb(40, 44, 52);
        txtInput2.BaseColor = Color.FromArgb(40, 44, 52);
        txtInput1.FocusedBaseColor = Color.FromArgb(40, 44, 52);
        txtInput2.FocusedBaseColor = Color.FromArgb(40, 44, 52);

        txtInput1.BorderColor = Color.FromArgb(40, 44, 52);
        txtInput2.BorderColor = Color.FromArgb(40, 44, 52);
        txtInput1.FocusedBorderColor = Color.FromArgb(40, 44, 52);
        txtInput2.FocusedBorderColor = Color.FromArgb(40, 44, 52);

        pnlCodeFrame.BackColor = Color.FromArgb(17, 18, 23);
        pnlFrame.BackColor = Color.FromArgb(40, 44, 52);

        btnShowAnswer.BaseColor = Color.FromArgb(17, 18, 23);
    } else {
        lblPracticeTask.ForeColor = Color.Black;
        lblTask.ForeColor = Color.DimGray;
        lblCode.ForeColor = Color.Black;

        txtInput1.ForeColor = Color.Black;
        txtInput2.ForeColor = Color.Black;
        txtInput1.FocusedForeColor = Color.Black;
        txtInput2.FocusedForeColor = Color.Black;

        txtInput1.BaseColor = Color.White;
        txtInput2.BaseColor = Color.White;
```

```
            txtInput1.FocusedBaseColor = Color.White;
            txtInput2.FocusedBaseColor = Color.White;

            txtInput1.BorderColor = Color.White;
            txtInput2.BorderColor = Color.White;
            txtInput1.FocusedBorderColor = Color.White;
            txtInput2.FocusedBorderColor = Color.White;

            pnlCodeFrame.BackColor = Color.FromArgb(231, 233, 235);
            pnlFrame.BackColor = Color.White;

            btnShowAnswer.BaseColor = Color.FromArgb(40, 42, 53);
        }

        if (!string.IsNullOrWhiteSpace(txtInput1.Text) || !string.IsNullOrWhiteSpace(txtInput2.Text)) {
            if (txtInput1.Text == "i = 0")
                txtInput1.BorderColor = Color.FromArgb(4, 170, 109);
            else
                txtInput1.BorderColor = Color.FromArgb(185, 74, 72);

            if (txtInput2.Text == "10")
                txtInput2.BorderColor = Color.FromArgb(4, 170, 109);
            else
                txtInput2.BorderColor = Color.FromArgb(185, 74, 72);
        }
    }
  }
}
```

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Guna.UI.Lib;

namespace trainer
{
    public partial class ucWhileDoWhileTask1 : UserControl
    {
        private static ucWhileDoWhileTask1 _instance;

        public static ucWhileDoWhileTask1 instance
        {
            get
            {
                if (_instance == null)
                    _instance = new ucWhileDoWhileTask1();
                return _instance;
            }
        }

        public ucWhileDoWhileTask1()
        {
            InitializeComponent();

            ThemeChecker.Start();

            GraphicsHelper.DrawLineShadow(pnlFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);
            GraphicsHelper.DrawLineShadow(pnlCodeFrame, Color.Black, 30, 18,
             Guna.UI.WinForms.VerHorAlign.VerticalLeft);

            lnklblNext.Visible = false;

            pctBoxStatus.Visible = false;
        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {
            pctBoxStatus.Visible = false;

            txtInput1.Text = "";
            txtInput2.Text = "";
            txtInput3.Text = "";

            txtInput1.BorderColor = Color.White;
```

```
    txtInput2.BorderColor = Color.White;
    txtInput3.BorderColor = Color.White;

    lnklblNext.Visible = false;

    btnSubmit.Visible = true;

    btnShowAnswer.Text = "Показати відповідь";
}

private void lnklblNext_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.Controls.Clear();

    ucWhileDoWhileTask2 task = new ucWhileDoWhileTask2();

    this.Controls.Add(task);
}

private void btnSubmit_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(txtInput1.Text) || !string.IsNullOrWhiteSpace(txtInput2.Text) ||
     !string.IsNullOrWhiteSpace(txtInput3.Text)) {
        if (txtInput1.Text == "while" && txtInput2.Text == "<" && txtInput3.Text == "i++"){
            pctBoxStatus.Visible = true;
            pctBoxStatus.Image = Properties.Resources.passed;

            lnklblNext.Visible = true;
        } else {
            pctBoxStatus.Visible = true;
            pctBoxStatus.Image = Properties.Resources.attention;

            lnklblNext.Visible = false;

            (new MsgBox(Type.WARNING, "Увага!", "Ви допустили помилку у коді. Будь-ласка перегляньте ще
     раз умову та код завдання.")).ShowDialog();
        }
    }
}

private void btnShowAnswer_Click(object sender, EventArgs e)
{
    if (btnShowAnswer.Text == "Показати відповідь") {
        txtInput1.Text = "while";
        txtInput2.Text = "<";
        txtInput3.Text = "i++";
```

```
        btnSubmit.Visible = false;

        btnShowAnswer.Text = "Скрити відповідь";
    } else if (btnShowAnswer.Text == "Скрити відповідь") {
        txtInput1.Text = "";
        txtInput2.Text = "";
        txtInput3.Text = "";

        btnSubmit.Visible = true;

        btnShowAnswer.Text = "Показати відповідь";
    }
}

private void ThemeChecker_Tick(object sender, EventArgs e)
{
    if (frmMain.isNight) {
        lblPracticeTask.ForeColor = Color.White;
        lblTask.ForeColor = Color.White;
        lblCode.ForeColor = Color.White;

        txtInput1.ForeColor = Color.White;
        txtInput2.ForeColor = Color.White;
        txtInput3.ForeColor = Color.White;
        txtInput1.FocusedForeColor = Color.White;
        txtInput2.FocusedForeColor = Color.White;
        txtInput3.FocusedForeColor = Color.White;

        txtInput1.BaseColor = Color.FromArgb(40, 44, 52);
        txtInput2.BaseColor = Color.FromArgb(40, 44, 52);
        txtInput3.BaseColor = Color.FromArgb(40, 44, 52);
        txtInput1.FocusedBaseColor = Color.FromArgb(40, 44, 52);
        txtInput2.FocusedBaseColor = Color.FromArgb(40, 44, 52);
        txtInput3.FocusedBaseColor = Color.FromArgb(40, 44, 52);

        txtInput1.BorderColor = Color.FromArgb(40, 44, 52);
        txtInput2.BorderColor = Color.FromArgb(40, 44, 52);
        txtInput3.BorderColor = Color.FromArgb(40, 44, 52);
        txtInput1.FocusedBorderColor = Color.FromArgb(40, 44, 52);
        txtInput2.FocusedBorderColor = Color.FromArgb(40, 44, 52);
        txtInput3.FocusedBorderColor = Color.FromArgb(40, 44, 52);

        pnlCodeFrame.BackColor = Color.FromArgb(17, 18, 23);
        pnlFrame.BackColor = Color.FromArgb(40, 44, 52);

        btnShowAnswer.BaseColor = Color.FromArgb(17, 18, 23);
    } else {
```

```
    lblPracticeTask.ForeColor = Color.Black;
    lblTask.ForeColor = Color.DimGray;
    lblCode.ForeColor = Color.Black;

    txtInput1.ForeColor = Color.Black;
    txtInput2.ForeColor = Color.Black;
    txtInput3.ForeColor = Color.Black;
    txtInput1.FocusedForeColor = Color.Black;
    txtInput2.FocusedForeColor = Color.Black;
    txtInput3.FocusedForeColor = Color.Black;

    txtInput1.BaseColor = Color.White;
    txtInput2.BaseColor = Color.White;
    txtInput3.BaseColor = Color.White;
    txtInput1.FocusedBaseColor = Color.White;
    txtInput2.FocusedBaseColor = Color.White;
    txtInput3.FocusedBaseColor = Color.White;

    txtInput1.BorderColor = Color.White;
    txtInput2.BorderColor = Color.White;
    txtInput3.BorderColor = Color.White;
    txtInput1.FocusedBorderColor = Color.White;
    txtInput2.FocusedBorderColor = Color.White;
    txtInput3.FocusedBorderColor = Color.White;

    pnlCodeFrame.BackColor = Color.FromArgb(231, 233, 235);
    pnlFrame.BackColor = Color.White;

    btnShowAnswer.BaseColor = Color.FromArgb(40, 42, 53);
}

If      (!string.IsNullOrWhiteSpace(txtInput1.Text)      ||      !string.IsNullOrWhiteSpace(txtInput2.Text)      ||
 !string.IsNullOrWhiteSpace(txtInput3.Text)) {
    if (txtInput1.Text == "while")
        txtInput1.BorderColor = Color.FromArgb(4, 170, 109);
    else
        txtInput1.BorderColor = Color.FromArgb(185, 74, 72);

    if (txtInput2.Text == "<")
        txtInput2.BorderColor = Color.FromArgb(4, 170, 109);
    else
        txtInput2.BorderColor = Color.FromArgb(185, 74, 72);

    if (txtInput3.Text == "i++")
        txtInput3.BorderColor = Color.FromArgb(4, 170, 109);
    else
        txtInput3.BorderColor = Color.FromArgb(185, 74, 72);
```

```
        }
      }
    }
  }
```