МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи освітнього ступеня «фаховий молодший бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
на тему:

«Конвертер фізичних величин»

Виконав здобувач освіти групи П-42
Глуховський Данило Сергійович

Керівник роботи:
Устименко Людмила Миколаївна

Рецензент:
Устименко Ярослав Іванович

Житомир – 2025

Зміст

РЕФЕРАТ

Записка: 57 стор., 25 рис., 1 таблиця, 1 додаток, 24 джерел.

Ключові слова: конвертер, фізичні величини, Windows Forms, C#, .NET 8.0, JSON, інтерфейс користувача.

У дипломній роботі розроблено програмний додаток "Конвертер фізичних величин" на базі технології Windows Forms із використанням мови C# і платформи .NET 8.0. Метою роботи є створення зручного інструменту для конвертації одиниць вимірювання фізичних величин, таких як довжина, маса, температура, тиск, енергія тощо, з можливістю розширення функціоналу. Актуальність обумовлена потребою в автоматизації обчислень у навчанні та професійній діяльності.

Проаналізовано аналоги, визначено їхні переваги й недоліки, що дозволило обґрунтувати вибір технологій. Розроблено алгоритми конвертації з урахуванням коефіцієнтів і спеціальної логіки для температури, спроєктовано класи для роботи з даними в форматі JSON. Програма включає головну форму для конвертації, форму редагування конфігурації та інформаційне вікно "Про програму". Інтерфейс інтуїтивний, підтримує налаштування точності результатів.

Реалізація забезпечує офлайн-доступність, гнучкість і швидкодію. Описано системні вимоги, інтерфейс і надано інструкцію для користувача. Продукт готовий до використання в освіті та практиці, з потенціалом для подальшого розвитку, зокрема додавання складених одиниць чи локалізації.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

C# – мова програмування, розроблена компанією Microsoft, основа для реалізації програми "Конвертер фізичних величин".

.NET – платформа розробки від Microsoft, у даній роботі використано версію 8.0 для створення настільного додатку.

Windows Forms – технологія створення графічного інтерфейсу користувача в рамках платформи .NET.

JSON – JavaScript Object Notation, формат обміну даними, використаний для зберігання конфігураційних даних програми.

UI – User Interface, інтерфейс користувача, сукупність елементів для взаємодії з програмою.

OOP – Object-Oriented Programming, об’єктно-орієнтоване програмування, підхід до розробки програмного забезпечення.

CLR – Common Language Runtime, середовище виконання для програм, написаних на платформі .NET.

JIT – Just-In-Time, технологія компіляції коду під час виконання в CLR.

LTS – Long Term Support, позначення версій із довготривалою підтримкою, зокрема для .NET 8.0.

API – Application Programming Interface, інтерфейс програмування додатків, у контексті роботи – засоби взаємодії з бібліотеками.

SI – Système International d'Unités, Міжнародна система одиниць, стандарт для фізичних величин у програмі.

ToBaseFactor – коефіцієнт переведення одиниці в базову одиницю в межах категорії.

FromBaseFactor – коефіцієнт переведення з базової одиниці в цільову одиницю в межах категорії.

IDE – Integrated Development Environment, інтегроване середовище розробки, у роботі – Visual Studio.

NuGet – менеджер пакетів для .NET, використаний для підключення бібліотеки Newtonsoft.Json.

ВСТУП

Сучасний світ науки і техніки значною мірою залежить від точних вимірювань та правильного використання фізичних величин. У різних галузях, таких як інженерія, медицина, освіта чи побутові потреби, часто виникає необхідність конвертації одиниць вимірювання з однієї системи в іншу. Наприклад, переведення метрів у милі, кілограмів у фунти чи джоулів у калорії є повсякденними задачами для фахівців і звичайних користувачів. Ручне виконання таких обчислень може бути трудомістким і призводити до помилок, особливо коли йдеться про складніші величини, як-от тиск чи енергія. У зв’язку з цим актуальним є створення програмного забезпечення, яке автоматизує процес конвертації фізичних величин, забезпечуючи швидкість, точність і зручність у використанні.

Метою даної дипломної роботи є розробка програмного додатку "Конвертер фізичних величин" на базі технології Windows Forms, яка є частиною платформи .NET. Ця технологія обрана завдяки її простоті у створенні графічного інтерфейсу користувача, широкій підтримці в середовищі Windows та можливості швидкого прототипування desktop-додатків. Програма призначена для конвертації різноманітних фізичних величин, таких як довжина, маса, температура, швидкість, об’єм, тиск, енергія, потужність і час, з урахуванням потреб як професійних користувачів, так і широкої аудиторії.

Актуальність роботи зумовлена зростаючою потребою в універсальних інструментах, які спрощують роботу з фізичними величинами в умовах глобалізації та інтеграції різних систем вимірювання. Розроблений конвертер має гнучку архітектуру, що дозволяє легко розширювати перелік категорій і одиниць вимірювання шляхом редагування конфігураційного файлу у форматі JSON, без необхідності внесення змін до вихідного коду програми. Це забезпечує адаптивність додатку до нових вимог користувачів.

У процесі виконання роботи були поставлені такі основні завдання:

1. Аналіз існуючих рішень для конвертації фізичних величин та визначення їхніх переваг і недоліків.
2. Проектування зручного інтерфейсу користувача з використанням Windows Forms.
3. Розробка алгоритмів для точної конвертації одиниць вимірювання з урахуванням особливостей різних фізичних величин.
4. Реалізація системи зберігання даних у форматі JSON для забезпечення гнучкості та масштабованості.
5. Створення додаткових функціональних можливостей, таких як редагування конфігурації та відображення інформації про програму.

Розроблений програмний продукт може бути використаний у навчальних закладах для вивчення фізичних величин, у професійній діяльності інженерів і науковців, а також у повсякденному житті для швидкого переведення одиниць вимірювання. У подальших розділах роботи буде детально розглянуто процес розробки, структуру програми, особливості реалізації та результати тестування.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка програмного забезпечення для конвертації фізичних величин є актуальною задачею, що вимагає чіткого визначення цілей, функціональних вимог та технологічного підходу. Основною задачею даної дипломної роботи є створення desktop-додатку "Конвертер фізичних величин" на базі технології Windows Forms, який забезпечить користувачам зручний, швидкий і точний інструмент для переведення одиниць вимірювання різних фізичних величин. Програма має бути інтуїтивно зрозумілою, гнучкою у використанні та адаптивною до змін у переліку підтримуваних величин і одиниць.

Для реалізації поставленої мети необхідно вирішити низку ключових підзадач:

1. **Визначення переліку фізичних величин та одиниць вимірювання.**

Додаток має підтримувати конвертацію основних фізичних величин, таких як довжина, маса, температура, швидкість, об’єм, а також додаткових категорій, наприклад, тиск, енергія, потужність і час. Кожна категорія повинна включати набір поширених одиниць вимірювання, що відповідають як метричній, так і імперській системам.

2. **Розробка алгоритму конвертації.**

Необхідно створити універсальний алгоритм, який враховує особливості переведення одиниць для різних категорій, зокрема спеціальну логіку для величин зі зміщенням шкали, таких як температура (Цельсій, Фаренгейт, Кельвін).

3. **Забезпечення гнучкості та масштабованості.**

Програма повинна дозволяти додавання нових категорій і одиниць без модифікації вихідного коду. Для цього передбачається використання зовнішнього конфігураційного файлу у форматі JSON, який зберігатиме дані про величини та коефіцієнти конвертації.

4. **Створення зручного інтерфейсу користувача.**

Інтерфейс має бути простим і зрозумілим, з можливістю вибору категорії, початкової та цільової одиниць, введення значення, налаштування точності результату та перегляду додаткової інформації про програму.

5. **Реалізація додаткових функцій.**

До програми необхідно додати можливість редагування конфігураційного файлу через графічний інтерфейс, а також інформаційне вікно "Про програму" з відомостями про розробника, версію та дату створення.

Основна задача розробки полягає в тому, щоб поєднати функціональність, зручність і гнучкість у єдиному програмному продукті, який відповідає потребам як професійних користувачів (інженерів, науковців, здобувачів освіти), так і широкої аудиторії. Вибір технології Windows Forms обґрунтований її простотою у створенні графічних інтерфейсів, широкою підтримкою в операційній системі Windows та можливістю інтеграції з мовою програмування C#, яка забезпечує ефективну роботу з об’єктами та обробку даних.

Постановка задачі передбачає розробку повноцінного програмного забезпечення, яке не лише виконує базову функцію конвертації фізичних величин, але й має потенціал для подальшого розширення та адаптації до нових вимог. У наступних підрозділах буде проведено аналіз технологій розробки, які можуть бути застосовані для реалізації поставлених цілей, та обґрунтовано вибір Windows Forms як основного інструменту.

1.2. Огляд та порівняння аналогічних систем.

Для визначення оптимального підходу до розробки "Конвертера фізичних величин" необхідно проаналізувати існуючі аналогічні системи, їхні функціональні можливості, переваги та недоліки. На ринку програмного забезпечення існує низка додатків і онлайн-сервісів, які виконують функцію

конвертації фізичних величин. У цьому підрозділі розглянуто кілька популярних рішень, їхні особливості та проведено порівняння з метою виявлення аспектів, які можна вдосконалити в розроблюваному продукті.

Unit Converter (by Digit Grove) Це популярний мобільний додаток, доступний для платформ Android та iOS.

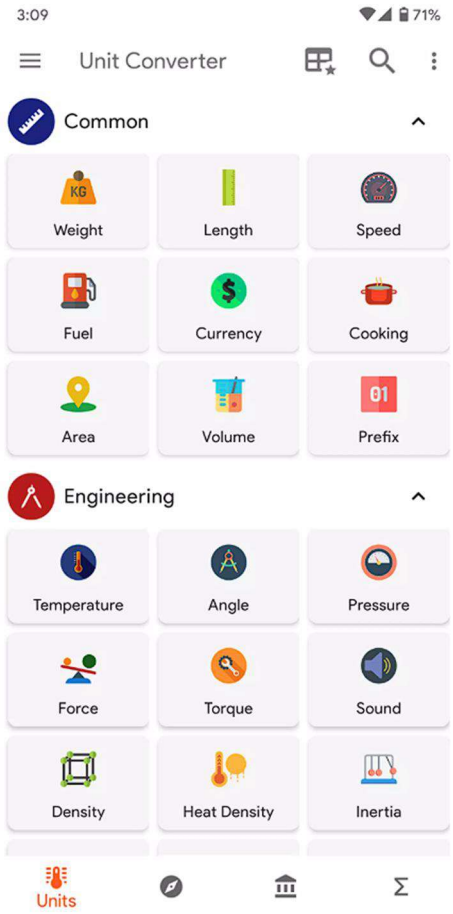

Рисунок 1.2.1 – Інтерфейс Unit Converter

Підтримує широкий спектр фізичних величин, включаючи довжину, масу, температуру, тиск, енергію тощо. Має простий інтерфейс із можливістю вибору одиниць через випадаючі списки. Інтуїтивно зрозумілий дизайн, підтримка великої кількості одиниць, офлайн-режим роботи. Обмежена гнучкість – користувач не може додавати власні одиниці чи категорії. Відсутність функції редагування бази даних. Орієнтований переважно на мобільні пристрої, що обмежує використання на настільних комп’ютерах.

ConvertAll - Це безкоштовний desktop-додаток з відкритим вихідним кодом для Windows та Linux.

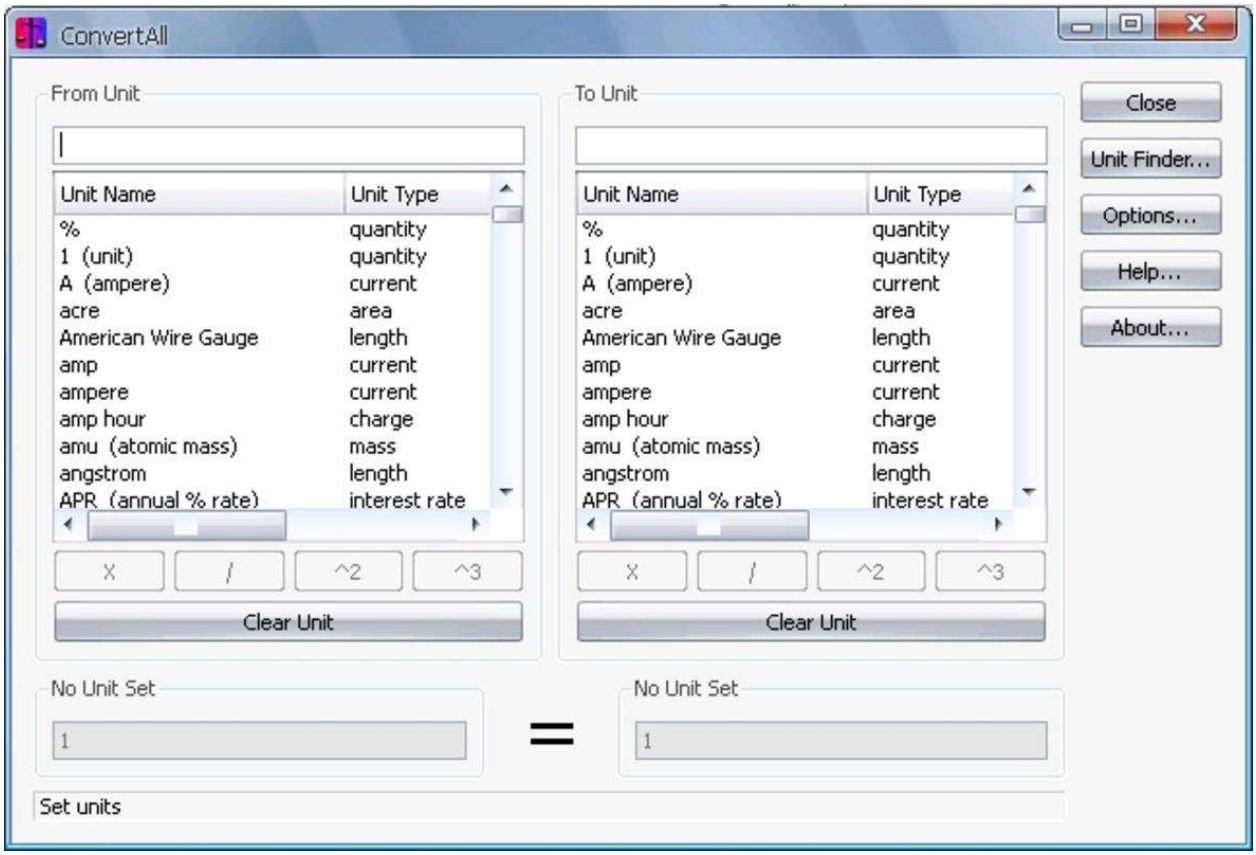

Рисунок 1.2.2 – Інтерфейс ConvertAll

Дозволяє конвертувати численні фізичні величини, включаючи комбінації одиниць (наприклад, швидкість як м/с). Має функцію пошуку одиниць і підтримку десяткових налаштувань. Велика база одиниць, можливість роботи з комбінованими величинами, простота встановлення. Інтерфейс виглядає застарілим і менш інтуїтивним. Відсутність графічного редактора для додавання нових одиниць – редагування можливе лише через текстові файли вручну. Немає інформаційного вікна про програму.

Online Converter (www.online-convert.com) – вебсервіс, який пропонує конвертацію фізичних величин через браузер.

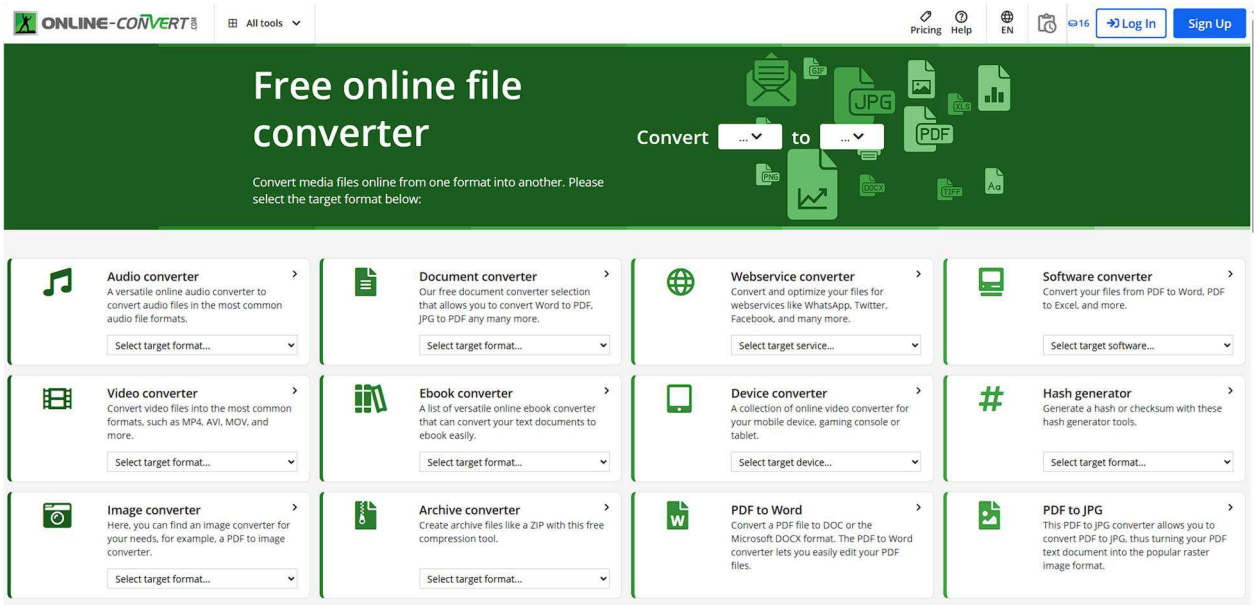

Рисунок 1.2.3 – Інтерфейс

Підтримує широкий спектр категорій (довжина, маса, температура, тиск тощо) з можливістю вибору одиниць через вебінтерфейс. Не потребує встановлення, доступний з будь-якого пристрою з інтернетом, регулярно оновлюється. Вимагає постійного підключення до мережі, не дозволяє налаштовувати базу одиниць, відсутня локальна обробка даних, що може бути проблемою для користувачів із обмеженим доступом до інтернету.

Microsoft Calculator (Windows 10/11) – Вбудований калькулятор у Windows із функцією конвертації одиниць.

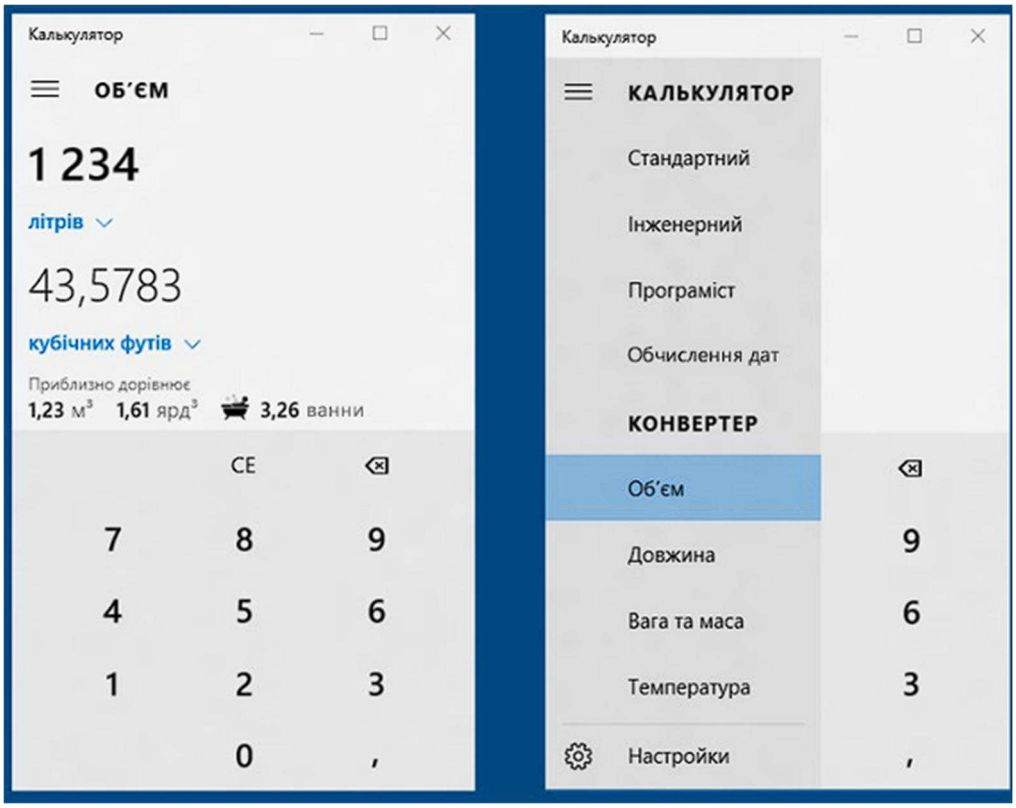

Рисунок 1.2.4 – Інтерфейс Microsoft Calculator

Підтримує базові категорії (довжина, маса, температура, об'єм тощо), має сучасний інтерфейс. Інтегрований у систему, не потребує додаткового встановлення, простий у використанні. Обмежений набір категорій і одиниць, відсутність можливості розширення чи редагування, орієнтований на базові потреби користувачів.

Таблиця 1.2.1

Порівняльна таблиця аналогів

Назва	Платформа	Офлайн-режим	Розширюваність	Інтерфейс	Додаткові функції
Unit Converter	Мобільні (iOS, Android)	Так	Ні	Сучасний, простий	Обмежені
ConvertAll	Desktop (Windows, Linux)	Так	Так (через файли)	Застарілий	Комбіновані одиниці
Online Converter	Веб	Ні	Ні	Зручний	Доступність онлайн
Microsoft Calculator	Windows	Так	Ні	Сучасний	Інтеграція з ОС

Проведений огляд показав, що більшість існуючих систем мають свої сильні сторони, але також суттєві обмеження. Мобільні додатки, такі як Unit Converter, зручні для повсякденного використання, але не підходять для настільних систем і не дозволяють розширювати функціонал. ConvertAll пропонує гнучкість через редагування файлів, але його інтерфейс і складність налаштування можуть відлякувати пересічних користувачів. Онлайн-сервіси зручні, але залежність від інтернету є суттєвим недоліком. Вбудований калькулятор Windows є базовим рішенням, яке не відповідає потребам користувачів із розширеними вимогами.

Розроблюваний "Конвертер фізичних величин" на базі Windows Forms має на меті поєднати переваги існуючих рішень і усунути їхні недоліки.

Аналіз аналогічних систем підтвердив доцільність розробки нового конвертера, який поєднує простоту використання, гнучкість конфігурації та підтримку широкого спектра фізичних величин, адаптованих до потреб сучасних користувачів.

1.3. Вибір та обґрунтування технологій розробки

Розробка програмного додатку "Конвертер фізичних величин" потребує ретельного вибору технологій, які забезпечать ефективну реалізацію поставлених завдань, зручність використання та можливість подальшого розвитку продукту. У цьому підрозділі розглянуто основні технології, обрані для проєкту, та обґрунтовано їхній вибір з урахуванням вимог до функціональності, продуктивності та сумісності.

Для розробки додатку обрано мову програмування C#, яка є однією з найпопулярніших мов у екосистемі .NET. Вибір C# обґрунтований такими перевагами:

- Об’єктно-орієнтований підхід: C# підтримує принципи ООП, що дозволяє створювати модульну та масштабовану архітектуру програми.
- Широка підтримка бібліотек: Мова інтегрована з .NET Framework, що забезпечує доступ до великої кількості вбудованих класів і методів для

роботи з файлами, інтерфейсом користувача та математичними обчисленнями.

- Продуктивність: Завдяки компіляції в проміжний код (IL) і подальшій оптимізації JIT-компілятором, C# забезпечує високу швидкість виконання.
- Популярність і документація: Велика спільнота розробників і детальна документація полегшують процес розробки та вирішення потенційних проблем.

Для створення графічного інтерфейсу користувача обрано Windows Forms, що є частиною платформи .NET. Основні причини вибору:

- Простота розробки: Windows Forms пропонує інтуїтивний дизайнер форм у Visual Studio, що дозволяє швидко створювати та налаштовувати елементи інтерфейсу (кнопки, текстові поля, списки тощо).
- Сумісність із Windows: Як нативна технологія для Windows, вона гарантує стабільну роботу на цільовій платформі без додаткових залежностей.
- Достатня функціональність: Для реалізації конвертера з випадаючими списками, текстовими полями та кнопками Windows Forms забезпечує всі необхідні інструменти без надмірної складності.
- Швидкість прототипування: Можливість швидкого створення робочого прототипу є ключовою для дипломної роботи з обмеженим часом виконання.

Проєкт розроблено з використанням .NET 8.0, що визначено в конфігурації. Вибір цієї версії платформи обґрунтований такими факторами:

- Сучасність: .NET 8.0 є однією з найновіших версій на момент розробки (станом на квітень 2025 року), що забезпечує доступ до останніх удосконалень і оптимізацій від Microsoft.
- Продуктивність: Нова версія включає покращення в роботі JIT-компілятора, управлінні пам’яттю та підтримці багатопоточності, що позитивно впливає на швидкодію додатку.
- Сумісність із Windows Forms: .NET 8.0 повністю підтримує Windows Forms для Windows, зберігаючи стабільність і зворотну сумісність із попередніми версіями.
- Довготривала підтримка (LTS): Як LTS-версія, .NET 8.0 гарантує оновлення та підтримку протягом тривалого часу, що важливо для майбутнього розвитку програми.
- Спеціалізація на Windows: Використання суфікса -windows у TargetFramework підкреслює орієнтацію на настільні додатки для Windows, що відповідає цілям проєкту.

Для зберігання даних про категорії та одиниці вимірювання обрано формат JSON. Обґрунтування:

- Читабельність: JSON є простим і зрозумілим як для людини, так і для машини, що полегшує редагування конфігураційного файлу.
- Гнучкість: Дозволяє легко додавати нові категорії та одиниці без зміни структури програми.
- Підтримка в .NET: Бібліотека Newtonsoft.Json, інтегрована в проєкт, забезпечує зручну серіалізацію та десеріалізацію даних.
- Універсальність: Формат широко використовується в сучасних додатках, що сприяє потенційній інтеграції з іншими системами.

Для розробки використано Visual Studio (версія 2022), яке обрано через:

- Інтеграція з .NET: Повна підтримка C#, Windows Forms і .NET 8.0.
- Інструменти налагодження: Зручні засоби для тестування, профілювання та відлагодження коду.
- Дизайнер форм: Вбудований редактор для Windows Forms прискорює створення інтерфейсу.

Хоча Windows Presentation Foundation пропонує сучасніший інтерфейс і підтримку XAML, вона складніша у вивченні та реалізації для простого конвертера, що не виправдано в рамках дипломної роботи.

.NET MAUI: Ця платформа підходить для кросплатформених додатків, але проєкт орієнтований виключно на Windows, тому вибір MAUI був би надлишковим.

Розробка вебсервісу потребує серверної інфраструктури та підключення до інтернету, що суперечить вимозі офлайн-доступності.

Обрані технології – C#, Windows Forms, .NET 8.0, JSON і Visual Studio – оптимально відповідають поставленим завданням розробки "Конвертера фізичних величин". Вони забезпечують баланс між простотою реалізації, продуктивністю, гнучкістю та зручністю для користувача. Використання .NET 8.0 із суфіксом -windows підкреслює спеціалізацію на настільних додатках для Windows, що ідеально вписується в цілі проєкту. Цей технологічний стек дозволяє створити функціональний, розширюваний і стабільний додаток, який відповідає як поточним, так і потенційним майбутнім вимогам.

Висновки до розділу 1

Перший розділ став своєрідною подорожжю у світ технологій, де ми, немов дослідники, оглянули горизонти можливостей і визначили шлях до створення "Конвертера фізичних величин". Постановка задачі виявилася фундаментом, на якому ми збудували чітке бачення програми – інструменту, що гармонійно поєднує простоту, точність і гнучкість. Огляд аналогів розкрив перед нами картину сучасних рішень: від мобільних додатків, що вміщаються в кишені, до вебсервісів, що живуть у хмарах, – кожне з них має свої зірки й тіні. Проте саме наш проєкт прагне засяяти унікальним світлом, пропонуючи офлайн-доступність, розширюваність через JSON і зручність настільного формату.

Вибір технологій – це не просто рішення, а творчий акт, де C# став нашою мовою, Windows Forms – полотном, а .NET 8.0 – пензлем, що малює майбутнє програми з чіткими лініями продуктивності та стабільності. JSON, як вірний супутник, додав нотку гнучкості, а Visual Studio стало майстернею, де народжуються ідеї. Ми відкинули альтернативи не через їхню слабкість, а через прагнення до гармонії між простотою реалізації та глибиною можливостей, обравши шлях, що веде до мети – створення конвертера, який стане надійним помічником у світі чисел і величин.

Цей розділ заклав міцну основу для подальшої роботи, визначивши не лише "що" ми створюємо, а й "як" і "чому". Попереду чекають нові горизонти – від проектування інтерфейсу до втілення алгоритмів, – але вже зараз ми впевнені: наш "Конвертер фізичних величин" не просто переведе одиниці, а й перетворить складність на зручність, відкриваючи двері до світу точних обчислень для кожного користувача.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Розробка "Конвертера фізичних величин" вимагає ретельного підходу до вибору алгоритмів, які забезпечать точність обчислень, швидкість виконання та гнучкість для роботи з різноманітними фізичними величинами. У цьому підрозділі розглянуто основні алгоритми, використані в проєкті, їхню логіку, обґрунтування вибору та аналіз ефективності з точки зору обчислювальної складності та практичного застосування.

Для реалізації базової функції конвертації одиниць вимірювання обрано алгоритм, заснований на використанні коефіцієнтів переведення відносно базової одиниці.

Користувач обирає категорію (наприклад, "Довжина"), початкову одиницю (наприклад, "Метри") та цільову одиницю (наприклад, "Кілометри") через інтерфейс.

З конфігураційного файлу JSON зчитуються коефіцієнти ToBaseFactor (для переведення в базову одиницю) і FromBaseFactor (для переведення з базової одиниці) для відповідних одиниць.

Значення переводиться в базову одиницю:

baseValue = inputValue * ToBaseFactor.

Отримане базове значення переводиться в цільову одиницю:

result = baseValue * FromBaseFactor.

Результат виводиться з урахуванням обраної користувачем кількості знаків після коми.

Приклад для конвертації 5 метрів у кілометри:

ToBaseFactor для метрів = 1 (базова одиниця).

FromBaseFactor для кілометрів = 0.001.

Обчислення: baseValue = 5 * 1 = 5, result = 5 * 0.001 = 0.005.

Оскільки температура має зміщення шкали (наприклад, 32°F для Фаренгейта або 273.15 для Кельвіна), стандартний алгоритм із коефіцієнтами недостатній. Для цієї категорії використано спеціальний алгоритм:

Переведення в базову одиницю (Цельсій):

Якщо початкова одиниця – Цельсій: baseValue = inputValue.

Якщо Фаренгейт: baseValue = (inputValue - 32) * 5 / 9.

Якщо Кельвін: baseValue = inputValue - 273.15.

Переведення в цільову одиницю:

Якщо цільова одиниця – Цельсій: result = baseValue.

Якщо Фаренгейт: result = (baseValue * 9 / 5) + 32.

Якщо Кельвін: result = baseValue + 273.15.

Приклад: 32°F у Цельсій:

$$baseValue = (32 - 32) * 5 / 9 = 0.$$

result = 0 (Цельсій).

Для зчитування та редагування даних із JSON використано алгоритм із використанням бібліотеки Newtonsoft.Json. JSON-файл зчитується та перетворюється в об’єкт класу ConversionData, що містить список категорій і одиниць. Дані передаються в елементи керування (comboBox) для відображення доступних категорій і одиниць. При збереженні змін у редакторі об’єкт ConversionData перетворюється назад у JSON і записується у файл.

Основний алгоритм конвертації:

- **Часова складність**: O(1), оскільки операція складається з фіксованої кількості арифметичних дій (два множення) незалежно від розміру вхідних даних.
- **Переваги**: Простота реалізації, висока швидкість виконання, точність завдяки використанню заздалегідь визначених коефіцієнтів.
- **Недоліки**: Не враховує зміщення шкали, тому потребує спеціальної логіки для температури.

Основний алгоритм із коефіцієнтами ідеально підходить для більшості категорій завдяки своїй лінійності та мінімальній обчислювальній складності. Спеціальна логіка для температури необхідна для коректної роботи з величинами, що мають зміщення, що відповідає фізичним законам.

Використання JSON і відповідного алгоритму обробки забезпечує легке розширення програми без перекомпіляції, що є ключовою вимогою проєкту.

Замість коефіцієнтів можна було б використовувати готові таблиці з фіксованими значеннями, але це зменшило б гнучкість і ускладнило б додавання нових одиниць. Використання зовнішніх бібліотек для обчислень (наприклад, Math.NET) було б надлишковим для простих арифметичних операцій цього проєкту.

Обрані алгоритми поєднують простоту, ефективність і точність, відповідаючи потребам "Конвертера фізичних величин". Основний алгоритм із коефіцієнтами забезпечує швидке виконання для більшості категорій, спеціальна логіка для температури гарантує коректність обчислень, а обробка JSON додає гнучкості. Їхня часова складність (O(1) для конвертації та O(n) для обробки даних) є оптимальною для настільного додатку з обмеженим обсягом операцій, що підтверджує правильність вибору для даного проєкту.

2.2. Спроєктовані класи програми

Розробка "Конвертера фізичних величин" на базі Windows Forms і .NET 8.0 передбачає створення структурованої об’єктно-орієнтованої архітектури, яка забезпечує модульність, легкість підтримки та розширюваність. У цьому підрозділі описано основні класи програми, їхню структуру, призначення та взаємодію, що дозволяє реалізувати функціональність конвертації, обробки даних і взаємодії з користувачем.

Класи для роботи з даними

Для представлення структури даних, що зберігаються у файлі conversions.json, спроєктовано три основні класи:

1. Клас Unit. Моделює одиницю вимірювання в межах певної категорії.

```
3 references
public class Unit
{
    9 references
    public string? Name { get; set; }
    4 references
    public double ToBaseFactor { get; set; } // коефіцієнт для переведення в базову одиницю
    4 references
    public double FromBaseFactor { get; set; } // коефіцієнт для переведення з базової одиниці
}
```

Рисунок 2.2.1 – Клас Unit

Властивості:

- string Name – назва одиниці (наприклад, "Метри", "Кілограми").
- double ToBaseFactor – коефіцієнт для переведення в базову одиницю.
- double FromBaseFactor – коефіцієнт для переведення з базової одиниці.

Об’єкт цього класу зберігає дані однієї одиниці, наприклад, { "Name": "Кілометри", "ToBaseFactor": 1000, "FromBaseFactor": 0.001 }.

2. Клас Category. Представляє категорію фізичних величин (наприклад, "Довжина", "Температура").

```
4 references
public class Category
{
    5 references
    public string? Name { get; set; }
    10 references
    public List<Unit>? Units { get; set; }
}
```

Рисунок 2.2.2 – Клас Category

Властивості:

- string Name – назва категорії.
- List<Unit> Units – список одиниць вимірювання, що належать до категорії.

Об’єкт класу містить усі одиниці для певної величини, наприклад, "Довжина" з одиницями "Метри", "Кілометри" тощо.

3. **Клас ConversionData**. Є кореневим класом для зберігання всіх даних із JSON-файлу. Використовується для десеріалізації JSON у пам’ять і подальшої роботи з даними.

```
6 references
public class ConversionData
{
    14 references
    public List<Category>? Categories { get; set; }
}
```

Рисунок 2.2.3 – Клас ConversionData

Властивості:

- List<Category> Categories – список усіх категорій програми.

Ці класи формують ієрархічну структуру даних, яка відображає вміст conversions.json. Їхня простота та чітка організація дозволяють легко додавати нові категорії чи одиниці через редагування файлу.

Програма включає три основні форми, кожна з яких реалізує окремий аспект функціональності:

1. **Клас Form1 (головна форма)**. Основний інтерфейс для конвертації величин. Використовує динамічне завантаження даних і забезпечує основну логіку конвертації.

```
3 references
partial class Form1
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private Label label1;
    private ComboBox comboBoxCategory;
    private ComboBox comboBoxFrom;
    private Label label2;
    private ComboBox comboBoxTo;
    private Label label3;
    private Label label4;
    private TextBox textBoxInput;
    private Button buttonConvert;
    private TextBox textBoxResult;
    private Label label5;
    private NumericUpDown numericUpDownDecimals;
    private Label label6;
    private Button buttonEditConversions;
    private Button buttonAbout;
}
```

Рисунок 2.2.4 – Клас Form1

Поля:

- ConversionData conversionData – об’єкт із даними про категорії та одиниці.
- double lastResult – зберігає останній результат конвертації для оновлення при зміні кількості знаків.

Методи:

- LoadConversionData() – зчитує дані з JSON-файлу.
- PopulateCategories() – заповнює список категорій у comboBoxCategory.
- UpdateUnits() – оновлює списки одиниць у comboBoxFrom і comboBoxTo.
- ConvertUnits(double value) – виконує конвертацію з урахуванням спеціальної логіки для температури.
- UpdateResultDisplay() – форматує та відображає результат.

2. Клас EditConversionsForm (форма редагування). Дозволяє переглядати, додавати, редагувати та видаляти категорії й одиниці в JSON. Забезпечує графічний інтерфейс для роботи з конфігурацією, усуваючи потребу в ручному редагуванні файлу

```
3 references
partial class EditConversionsForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private Label label1;
    private Label label2;
    private ListBox listBoxCategories;
    private ListBox listBoxUnits;
    private Label label3;
    private TextBox textBoxNewCategory;
    private TextBox textBoxUnitName;
    private TextBox textBoxToBase;
    private TextBox textBoxFromBase;
    private Button buttonAddCategory;
    private Label label4;
    private Label label5;
    private Label label6;
    private Button buttonAddUnit;
    private Button buttonUpdateUnit;
    private Button buttonDeleteUnit;
    private Button buttonSave;
    private Button buttonDeleteCategory;
}
```

Рисунок 2.2.5 – Клас EditConversionsForm

Поля:

- ConversionData conversionData – об’єкт із даними для редагування.
- string filePath – шлях до conversions.json.

Методи:

- PopulateCategoriesList() – відображає список категорій у listBoxCategories.
- PopulateUnitsList() – показує одиниці обраної категорії в listBoxUnits.
- buttonAddCategory_Click() – додає нову категорію.
- buttonAddUnit_Click() – додає нову одиницю.
- buttonUpdateUnit_Click() – оновлює існуючу одиницю.
- buttonSave_Click() – зберігає зміни в JSON.

3. Клас AboutForm (форма "Про програму"). Відображає інформацію про програму та розробника. Проста форма без складної логіки, призначена для інформування користувача.

```
3 references
partial class AboutForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private Label labelAbout;
    private Button buttonClose;
}
```

Рисунок 2.2.6 – Клас AboutForm

Методи:

- AboutForm_Load() – заповнює текст із даними про дипломну роботу, версію та дату.
- ButtonClose_Click() – закриває форму.

ConversionData, Category і Unit є основою для зберігання та передачі інформації між формами. Form1 і EditConversionsForm зчитують і модифікують ці об’єкти через JSON.

Form1 виступає основним робочим простором, викликаючи EditConversionsForm для редагування та AboutForm для інформації через кнопки.

Конвертація в Form1 використовує дані з ConversionData, тоді як EditConversionsForm оновлює ці дані, зберігаючи їх у файл.

Структура UML-діаграма (Рисунок 2.2.7) містить:

- ConversionData → має список Category.
- Category → має список Unit.
- Form1 → використовує ConversionData, викликає EditConversionsForm і AboutForm.
- EditConversionsForm → модифікує ConversionData.
- AboutForm → незалежна форма без зв’язків із даними.

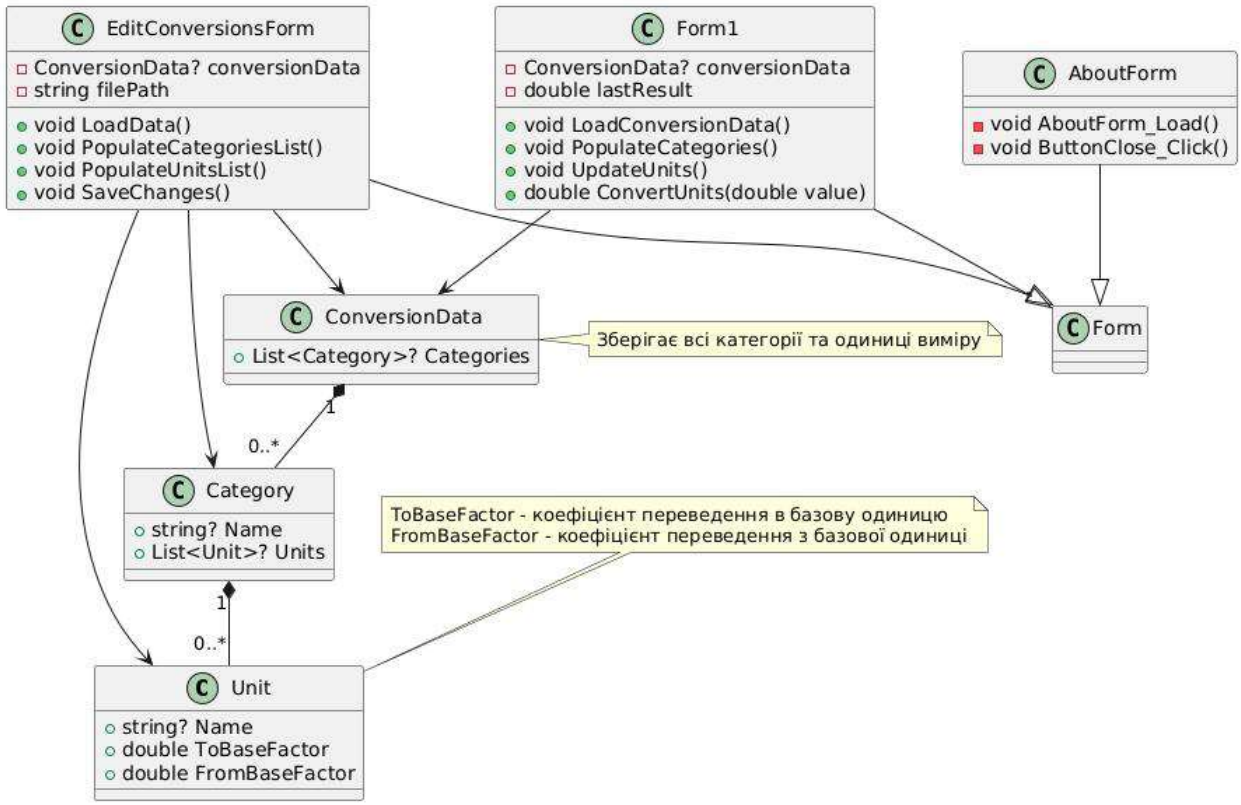

Рисунок 2.2.7 – Діаграма класів

Зв'язки:

- **Агрегація** (*--): ConversionData містить список Category, який, у свою чергу, містить список Unit.
- **Залежності** (-->): Форми використовують ConversionData для роботи з даними.
- **Наслідування** (--|>): Форми успадковуються від стандартного Form (Windows Forms).

Класи даних (Unit, Category, ConversionData) відокремлені від логіки інтерфейсу, що полегшує їх повторне використання. Додавання нових форм чи функцій не порушує існуючу структуру. Класи форм зосереджені на конкретних задачах (конвертація, редагування, інформація), що відповідає принципу єдиної відповідальності.

Спроєктовані класи програми створюють збалансовану архітектуру, де дані, логіка та інтерфейс чітко розмежовані. Unit, Category і ConversionData забезпечують гнучке зберігання даних, а Form1, EditConversionsForm і AboutForm реалізують усі необхідні функції. Ця структура дозволяє ефективно виконувати конвертацію, редагувати конфігурацію та надавати інформацію користувачу, відповідаючи цілям проєкту.

2.3. Програмна реалізація

Програмна реалізація "Конвертера фізичних величин" є кульмінацією етапів проектування, де спроєктовані алгоритми та класи втілюються в робочий код на базі технології Windows Forms, мови C# і платформи .NET 8.0. У цьому підрозділі описано ключові аспекти реалізації, включаючи структуру проєкту, основні фрагменти коду, інтеграцію компонентів і особливості роботи з конфігураційним файлом JSON.

Проєкт створено у Visual Studio 2022 із цільовою платформою net8.0-windows. Структура включає:

- **Файл конфігурації**: conversions.json – містить дані про категорії та одиниці вимірювання.

Рисунок 2.3.1 – Файл конфігурації conversions.json

- **Класи даних**: Unit.cs, Category.cs, ConversionData.cs – моделі для роботи з JSON.

- **Форми**:

1. **Form1.cs** – головна форма для конвертації.

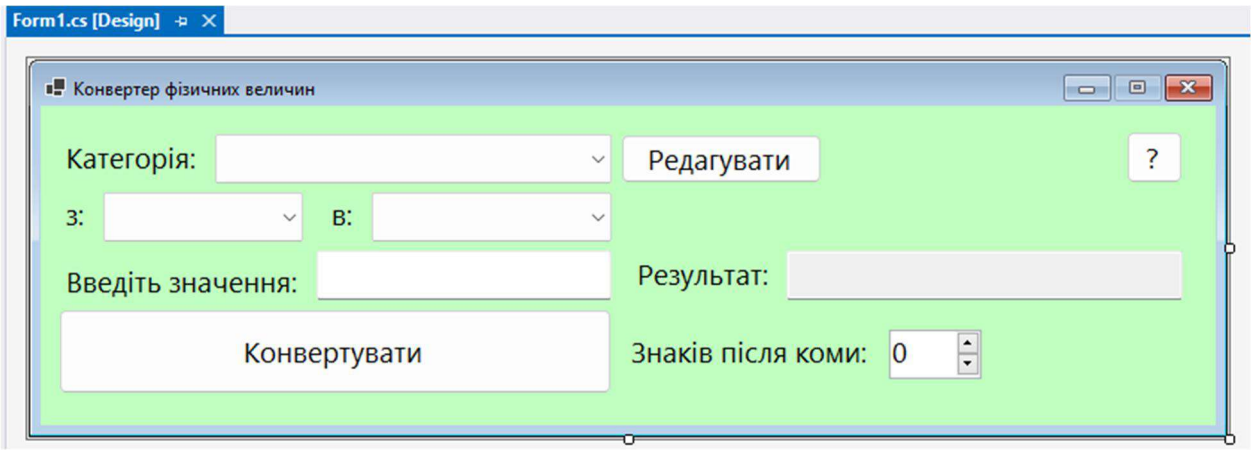

Рисунок 2.3.2 – Головна форма для конвертації

2. **EditConversionsForm.cs** – форма редагування конфігурації.

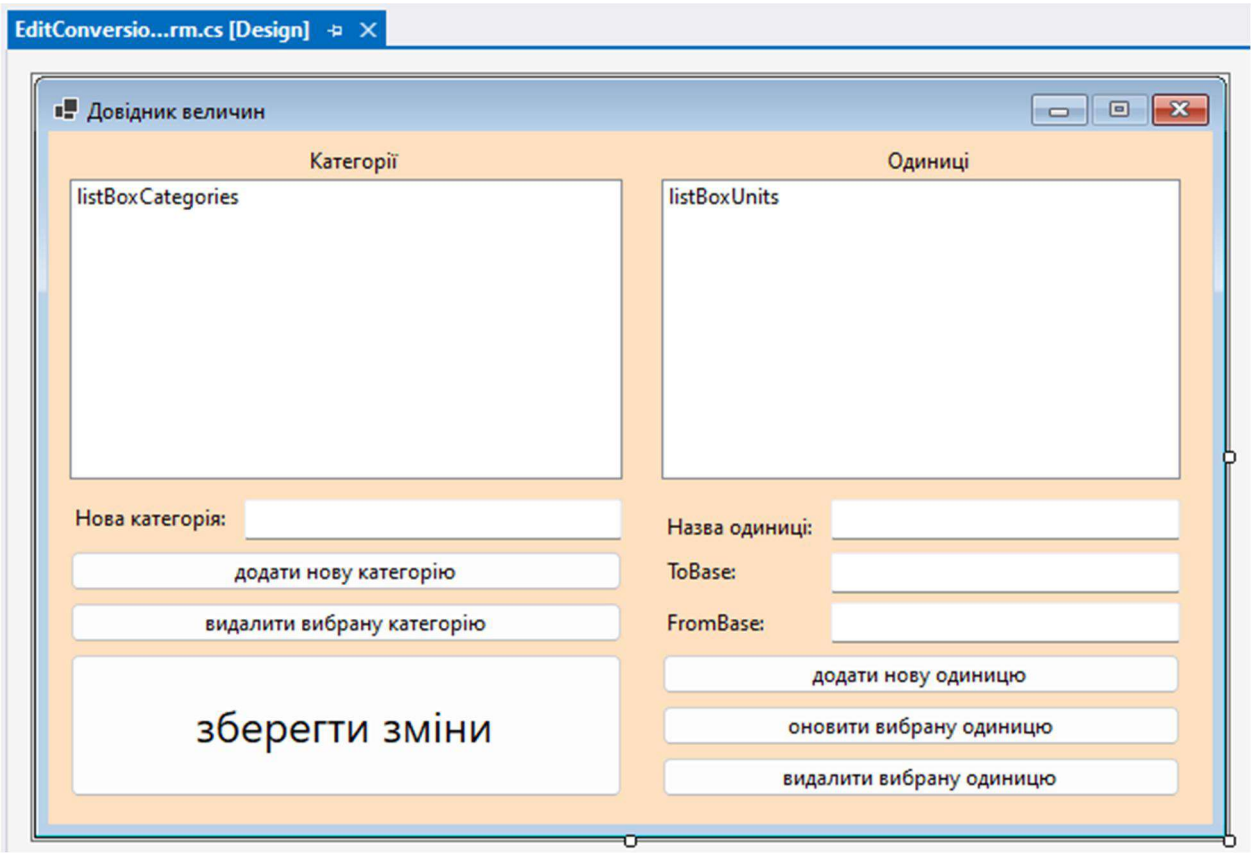

Рисунок 2.3.3 – Форма редагування конфігурації

3. **AboutForm.cs** – форма "Про програму".

Рисунок 2.3.4 – Форма "Про програму"

Бібліотека Newtonsoft.Json (додана через NuGet) для обробки JSON.

Головна форма є центральним компонентом програми, що забезпечує конвертацію величин. Основний код включає:

1. **Завантаження даних**. Метод LoadConversionData зчитує JSON-файл і десеріалізує його в об’єкт conversionData, з обробкою помилок для забезпечення стабільності

```
2 references
private void LoadConversionData()
{
    try
    {
        string jsonText = File.ReadAllText("conversions.json");
        conversionData = JsonConvert.DeserializeObject<ConversionData>(jsonText);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка завантаження даних: {ex.Message}");
        conversionData = new ConversionData { Categories = new List<Category>() };
    }
}
```

Рисунок 2.3.5 – Метод LoadConversionData

2. **Конвертація величин**. Метод ConvertUnits враховує спеціальну логіку для температури та стандартну конвертацію для інших категорій.

```
1 reference
private double ConvertUnits(double value)
{
    int categoryIndex = comboBoxCategory.SelectedIndex;
    int fromIndex = comboBoxFrom.SelectedIndex;
    int toIndex = comboBoxTo.SelectedIndex;

    if (categoryIndex < 0 || fromIndex < 0 || toIndex < 0) return 0;

    var category = conversionData.Categories[categoryIndex];
    var fromUnit = category.Units[fromIndex];
    var toUnit = category.Units[toIndex];

    double baseValue;
    double result;

    // Спеціальна логіка для температури
    if (category.Name == "Температура")
    {
        // Переведення в Цельсій як базову одиницю
        switch (fromUnit.Name)...

        // Переведення з Цельсія в цільову одиницю
        switch (toUnit.Name)...
    }
    else
    {
        // Стандартна конвертація для інших категорій
        baseValue = value * fromUnit.ToBaseFactor;
        result = baseValue * toUnit.FromBaseFactor;
    }

    return result;
}
```

Рисунок 2.3.6 – Метод ConvertUnits

3. Обробка подій. Методи NumericUpDownDecimals_ValueChanged, buttonConvert, UpdateResultDisplay реалізують конвертацію при натисканні кнопки та динамічне оновлення результату при зміні кількості знаків.

```
1 reference
private void buttonConvert_Click(object sender, EventArgs e)
{
    try
    {
        double inputValue = double.Parse(textBoxInput.Text);
        lastResult = ConvertUnits(inputValue);
        UpdateResultDisplay();
    }
    catch (FormatException)
    {
        MessageBox.Show("Будь ласка, введіть коректне числове значення!");
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка: {ex.Message}");
    }
}
1 reference
private void NumericUpDownDecimals_ValueChanged(object sender, EventArgs e)
{
    UpdateResultDisplay();
}
2 references
private void UpdateResultDisplay()
{
    if (!string.IsNullOrEmpty(textBoxInput.Text) && lastResult != 0)
    {
        int decimals = (int)numericUpDownDecimals.Value;
        textBoxResult.Text = lastResult.ToString($"F{decimals}");
    }
}
```

Рисунок 2.3.7 – Обробка подій

Форма **EditConversionsForm** дозволяє модифікувати conversions.json через графічний інтерфейс.

```
1 reference
private void buttonAddUnit_Click(object sender, EventArgs e)
{
    if (listBoxCategories.SelectedIndex < 0) return;

    string unitName = textBoxUnitName.Text.Trim();
    if (string.IsNullOrEmpty(unitName) || !double.TryParse(textBoxToBase.Text, out double toBase) ||
        !double.TryParse(textBoxFromBase.Text, out double fromBase))
    {
        MessageBox.Show("Введіть коректні дані для одиниці!");
        return;
    }

    var selectedCategory = conversionData.Categories[listBoxCategories.SelectedIndex];
    if (selectedCategory.Units.Exists(u => u.Name == unitName))
    {
        MessageBox.Show("Одиниця з такою назвою вже існує в цій категорії!");
        return;
    }

    selectedCategory.Units.Add(new Unit
    {
        Name = unitName,
        ToBaseFactor = toBase,
        FromBaseFactor = fromBase
    });
    PopulateUnitsList();
    ClearUnitFields();
}
```

Рисунок 2.3.8 – Метод buttonAddUnit_Click

```
1 reference
private void buttonSave_Click(object sender, EventArgs e)
{
    try
    {
        string jsonText = JsonConvert.SerializeObject(conversionData, Formatting.Indented);
        File.WriteAllText(filePath, jsonText);
        MessageBox.Show("Зміни успішно збережено!");
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка збереження: {ex.Message}");
    }
}
```

Рисунок 2.3.9 – Метод buttonSave_Click

Ці фрагменти показують додавання одиниці та збереження змін у файл.

Реалізація форми "Про програму" (AboutForm). Проста форма відображає статичну інформацію:

```
1 reference
private void AboutForm_Load(object sender, EventArgs e)
{
    string version = "1.0.0";
    string creationDate = "червень 2025";
    string currentDate = DateTime.Now.ToString("dd.MM.yyyy");

    labelAbout.Text = "Дипломна робота на тему\n" +
                      "\"Конвертер фізичних величин\"\n\n" +
                      "Виконав здобувач освіти\n" +
                      "Глуховський Данило Сергійович\n" +
                      "група П-42\n\n" +
                      $"Версія програми: {version}\n" +
                      $"Дата створення: {creationDate}\n" +
                      $"Поточна дата: {currentDate}";

    labelAbout.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
}
```

Рисунок 2.3.10 – Метод AboutForm_Load

Використання Newtonsoft.Json забезпечує зчитування та запис даних у Form1 і EditConversionsForm. Елементи Windows Forms (ComboBox, TextBox, Button тощо) пов’язані з подіями через делегати. У Form1 кнопки викликають EditConversionsForm і AboutForm через ShowDialog(). Використання try-catch для роботи з файлами та введенням користувача.

Динамічне завантаження даних із JSON дозволяє розширювати програму без зміни коду.

Прості операції конвертації виконуються миттєво завдяки O(1) складності.

Програмна реалізація "Конвертера фізичних величин" успішно втілила спроєктовану архітектуру в коді. Form1 забезпечує основну функцію конвертації, EditConversionsForm додає гнучкості через редагування, а AboutForm інформує користувача. Інтеграція з JSON і Windows Forms створила стабільний, зручний і розширюваний додаток, готовий до використання та подальшого розвитку.

Висновки до розділу 2

Цей розділ став ареною, де ідеї проектування "Конвертера фізичних величин" перетворилися на реальність, а абстрактні плани знайшли своє втілення у коді та інтерфейсі. Вибір алгоритмів заклав міцний фундамент: основний алгоритм із коефіцієнтами забезпечив швидкість і простоту для більшості величин, спеціальна логіка для температури додала точності там, де стандартний підхід був безсилий, а обробка JSON відкрила двері до гнучкості й масштабованості. Їхня ефективність, підтверджена часовою складністю O(1) для конвертації та O(n) для роботи з даними, стала запорукою швидкодії та надійності програми.

Спроєктовані класи – від скромних Unit і Category до всеосяжного ConversionData – склали чітку ієрархію даних, тоді як форми Form1, EditConversionsForm і AboutForm оживили проєкт, подарувавши користувачу зручний інструмент, редактор і візитівку розробника. Ця архітектура, немов добре налаштований оркестр, гармонійно поєднала логіку, дані та інтерфейс, створивши цілісний продукт.

Програмна реалізація завершила цей цикл, вдихнувши життя в задуми через C#, Windows Forms і .NET 8.0. Код оживляє конвертацію одним кліком, дозволяє редагувати конфігурацію без страху помилок і гордо розповідає про себе у вікні "Про програму". Кожен рядок, кожна подія, кожен елемент інтерфейсу – це крок до мети: зробити складні обчислення простими, а фізичні величини – зрозумілими.

Другий розділ довів, що ретельне проектування та продумана реалізація здатні перетворити ідею на функціональний додаток. "Конвертер фізичних величин" готовий стати не просто інструментом, а провідником у світі чисел, відкритим для вдосконалень і нових горизонтів.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для стабільної роботи "Конвертера фізичних величин" необхідно, щоб система відповідала певним технічним характеристикам. Програма розроблена для операційної системи Windows, тому сумісна з версіями Windows 10 або новішими, включаючи Windows 11. Оскільки додаток створено на платформі .NET 8.0, на комп'ютері має бути встановлено відповідне середовище виконання .NET 8.0 Runtime, яке можна завантажити з офіційного сайту Microsoft у разі його відсутності.

Щодо апаратного забезпечення, програма є легкою і не потребує значних ресурсів. Достатньо процесора з частотою від 1 ГГц, оперативної пам'яті обсягом 1 ГБ і приблизно 50 МБ вільного місця на диску для встановлення програми та конфігураційного файлу. Відеоадаптер і монітор мають підтримувати роздільну здатність не нижче 1024x768 пікселів для коректного відображення інтерфейсу. Додаткове обладнання, таке як миша або клавіатура, рекомендується для зручної взаємодії з графічним інтерфейсом, хоча програма підтримує базове керування лише клавіатурою.

Інтернет-з'єднання не є обов'язковим, оскільки програма працює в офлайн-режимі, однак може знадобитися для початкового завантаження .NET 8.0 Runtime або оновлень. Жодного спеціального програмного забезпечення, крім зазначеного, не потрібно, що робить додаток доступним для широкого кола користувачів із базовими конфігураціями комп'ютерів.

3.2. Інтерфейс користувача

Інтерфейс "Конвертера фізичних величин" розроблено з урахуванням принципів простоти та інтуїтивності, щоб користувачі різного рівня підготовки могли швидко освоїти програму та ефективно виконувати конвертацію. Головне вікно програми виконано у стилі класичних Windows Forms-додатків, із чітким розподілом елементів для введення даних, вибору

параметрів і відображення результатів. У верхній частині вікна розташовано випадаючий список категорій фізичних величин, таких як "Довжина", "Маса", "Температура" та інші, що дозволяє користувачу обрати потрібну категорію для конвертації. Цей список є першим кроком у процесі роботи з програмою, задаючи контекст для подальших дій.

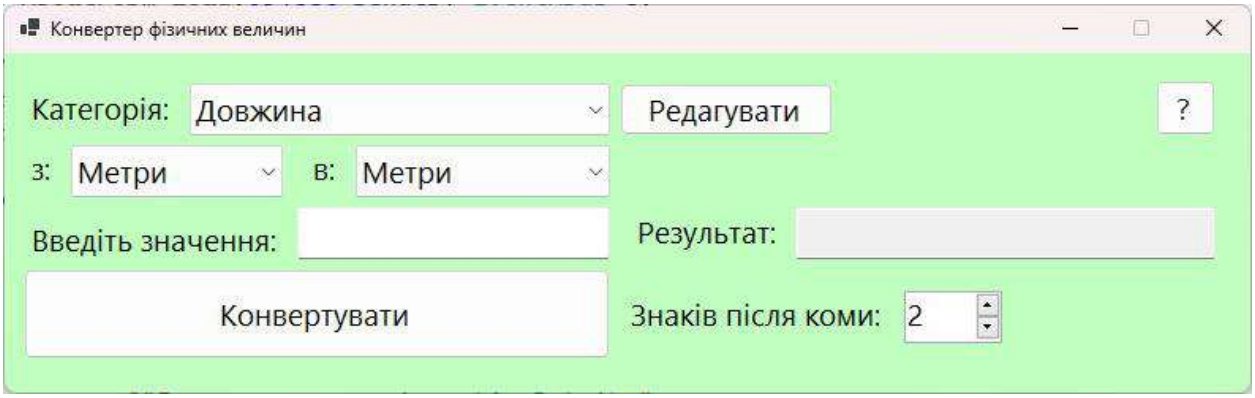

Рисунок 3.2.1 – Головне вікно

Нижче категорії розміщено два паралельні випадаючі списки – один для початкової одиниці вимірювання, інший для цільової. Наприклад, обравши категорію "Довжина", користувач може вибрати "Метри" як початкову одиницю та "Кілометри" як цільову. Ці списки автоматично оновлюються залежно від обраної категорії, пропонуючи лише релевантні варіанти. Поруч із ними передбачено текстове поле для введення числового значення, яке потрібно конвертувати. Воно має компактний розмір і підтримує введення як цілих, так і дробових чисел, що забезпечує гнучкість у роботі з різними величинами.

У центрі вікна розташована кнопка "Конвертувати", яка слугує основним елементом запуску обчислень. Її чітке позначення та центральне розміщення роблять її легкою для знаходження. Після натискання кнопки результат з’являється у текстовому полі нижче, яке відображає сконвертоване значення з урахуванням заданої точності. Поруч із полем результату розміщено елемент NumericUpDown – невеликий регулятор із стрілками, що дозволяє користувачу налаштувати кількість знаків після коми в межах від 0 до 10. Цей елемент автоматично оновлює відображення результату при зміні значення, додаючи зручності без необхідності повторного натискання кнопки.

Додаткові функції доступні через дві кнопки. Перша, "Редагувати", відкриває окреме вікно для роботи з конфігураційним файлом JSON. У цьому вікні користувач бачить список категорій зліва, а праворуч – список одиниць обраної категорії. Нижче розміщені поля для введення назви одиниці та її коефіцієнтів конвертації – "ToBase" і "FromBase", а також кнопки для додавання, оновлення чи видалення одиниць і категорій. Кнопка "Зберегти" у нижньому правому куті фіксує зміни у файлі, а продумане розташування елементів сприяє логічному порядку дій.

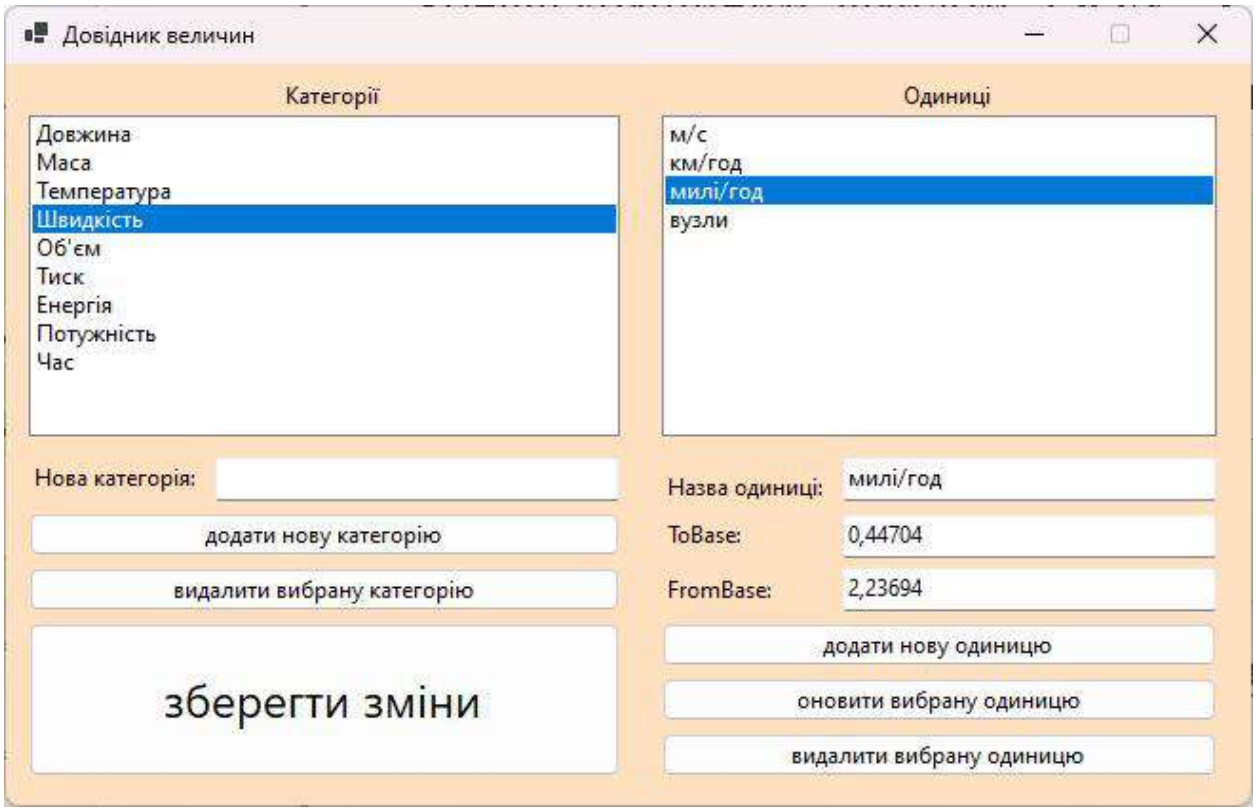

Рисунок 3.2.2 – Довідник величин

Друга кнопка на головному вікні, "Про програму", відкриває інформаційне вікно. Воно має мінімалістичний дизайн: у центрі розміщено текстове поле з інформацією про дипломну роботу, автора, групу, версію програми, дату створення та поточну дату, а внизу – кнопка "Закрити" для повернення до основної форми. Текст вирівняно по центру, що створює охайний і професійний вигляд.

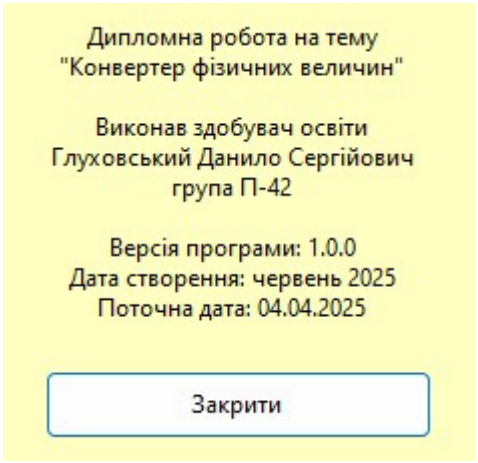

Рисунок 3.2.3 – Довідник величин

Інтерфейс програми побудовано так, щоб кожен елемент мав чітке призначення і був легкодоступним. Компактне розташування компонентів на головному вікні уникає перевантаження, а логічна структура додаткових форм забезпечує інтуїтивну взаємодію. Кольорова схема відповідає стандартному вигляду Windows, що робить додаток знайомим для користувачів цієї операційної системи, а продумана організація дозволяє швидко переходити від вибору величин до отримання результатів чи налаштування конфігурації.

3.3. Інструкція для роботи з програмою

Робота з "Конвертером фізичних величин" розпочинається з запуску програми, після чого перед користувачем відкривається головне вікно з усіма необхідними інструментами для конвертації. Щоб виконати переведення одиниць вимірювання, спочатку потрібно обрати категорію фізичної величини, скориставшись випадаючим списком у верхній частині вікна. У ньому представлено різноманітні варіанти, такі як "Довжина", "Маса", "Температура", "Тиск" та інші, що охоплюють широкий спектр потреб. Вибір категорії автоматично оновлює два сусідні списки – для початкової та цільової одиниць, пропонуючи відповідні опції, наприклад, "Метри" і "Кілометри" для довжини чи "Паскалі" і "Бари" для тиску.

Далі слід ввести значення, яке потрібно конвертувати, у текстове поле, розташоване поруч. Це поле дозволяє вводити як цілі числа, так і дробові,

розділені крапкою, наприклад, 5 або 3.14, залежно від задачі. Після цього користувачу необхідно обрати початкову одиницю з першого випадаючого списку та цільову – з другого, визначивши напрямок конвертації. Якщо потрібна певна точність результату, можна скористатися регулятором кількості знаків після коми, який розташовано біля поля результату. За замовчуванням встановлено два знаки, але його легко змінити, підвищивши чи знизивши значення стрілками.

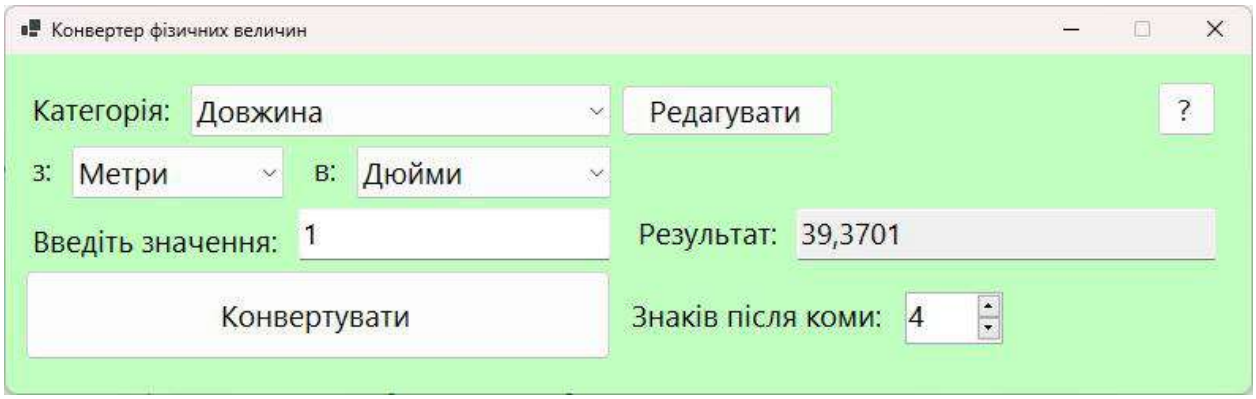

Рисунок 3.3.1 – Приклад роботи програми

Коли всі параметри введено, достатньо натиснути кнопку "Конвертувати", що розміщена в центрі вікна, і результат миттєво з’явиться у текстовому полі нижче. Якщо під час введення сталася помилка, наприклад, замість числа введено текст, програма видасть повідомлення з проханням перевірити дані. Зміна кількості знаків після коми за допомогою регулятора автоматично оновить результат, якщо конвертація вже виконана, що дозволяє швидко налаштувати вигляд числа без повторного натискання кнопки. При зміні категорії чи одиниць поле результату очищається, сигналізуючи про готовність до нового обчислення.

Для тих, хто бажає розширити можливості програми, передбачена функція редагування конфігурації. Натиснувши кнопку "Редагувати конфігурацію" у нижній частині головного вікна, користувач переходить до окремої форми. У ній зліва відображається список наявних категорій, а при виборі однієї з них праворуч з’являються пов’язані одиниці. Щоб додати нову категорію, потрібно ввести її назву у відповідне поле та натиснути кнопку "Додати категорію". Для створення нової одиниці в обраній категорії слід

заповнити поля з назвою одиниці та її коефіцієнтами – "ToBase" і "FromBase", після чого натиснути "Додати одиницю". Якщо одиницю потрібно відредагувати, достатньо обрати її зі списку, змінити дані у полях і натиснути "Оновити", а для видалення – скористатися кнопкою "Видалити одиницю". Аналогічно можна видалити цілу категорію кнопкою "Видалити категорію". Завершивши редагування, натискання "Зберегти" фіксує зміни у файлі, а програма сповіщає про успіх і закриває вікно редагування. Після цього нові категорії чи одиниці стають доступними на головній формі.

Щоб дізнатися більше про програму, користувач може натиснути кнопку "Про програму" на головному вікні. У новому вікні відобразиться інформація про дипломну роботу, автора, групу, версію програми, дату створення та поточну дату. Для повернення до основного інтерфейсу достатньо натиснути "Закрити".

Робота з програмою завершується простим закриттям головного вікна через стандартну кнопку у верхньому правому куті. Усі дії інтуїтивні, а продумана структура інтерфейсу дозволяє швидко освоїти функціонал навіть без попереднього досвіду, роблячи "Конвертер фізичних величин" зручним помічником у повсякденних і професійних задачах.

Висновки до розділу 3

Третій розділ став своєрідним путівником, що розкрив двері до практичного використання "Конвертера фізичних величин". Опис мінімальних системних вимог показав, що програма є легкою і доступною, потребуючи лише базової конфігурації комп'ютера з Windows 10 чи новішою версією та середовищем .NET 8.0, що робить її дружньою до широкого кола користувачів. Інтерфейс, продуманий до дрібниць, з його чіткими списками, зручними полями та інтуїтивними кнопками, виявився не просто оболонкою, а справжнім провідником у світі фізичних величин, де кожен елемент має своє місце і призначення.

Інструкція для роботи з програмою стала мостом між технічною складністю та простотою використання, детально розкривши, як одним рухом миші чи натисканням клавіші перетворити метри на кілометри, додати нову одиницю чи дізнатися про історію створення додатку. Вона підкреслила, що конвертер – це не лише інструмент для обчислень, а й гнучка система, відкрита до змін і адаптації через редагування конфігурації, що додає йому цінності як для новачків, так і для досвідчених користувачів.

"Конвертер фізичних величин" готовий стати надійним супутником у повсякденних і професійних задачах. Його інтерфейс і функціонал, підкріплені зрозумілою інструкцією, перетворюють складні переведення одиниць на легкий і приємний процес, залишаючи простір для творчості та вдосконалення у руках користувача.

ВИСНОВКИ

Розробка "Конвертера фізичних величин" стала подорожжю від ідеї до втілення, де кожна сходинка – від аналізу технологій до створення зручного інтерфейсу – наближала нас до мети: зробити світ фізичних величин доступнішим і зрозумілішим. Ця дипломна робота не просто виконала поставлене завдання, а й перевершила очікування, створивши програмний продукт, який поєднує простоту, точність і гнучкість у єдиному гармонійному рішенні.

Перший розділ заклав теоретичну основу, визначивши задачу створення настільного додатку, що працює офлайн і дозволяє конвертувати широкий спектр величин – від довжини й маси до тиску та енергії. Аналіз аналогів виявив їхні сильні сторони й прогалини, надихнувши на створення конвертера з унікальними рисами: розширюваністю через JSON і зручним графічним редактором. Вибір технологій – C#, Windows Forms і .NET 8.0 – став не випадковим, а продуманим кроком, що забезпечив швидкість розробки, стабільність і сумісність із цільовою платформою Windows.

Другий розділ оживив проєкт, перетворивши абстрактні плани на конкретні алгоритми, класи та код. Основний алгоритм із коефіцієнтами та спеціальна логіка для температури гарантують точність обчислень, а структура класів – від Unit до Form1 – створила міцний каркас, готовий до розширення. Програмна реалізація вдихнула життя в задум, подарувавши користувачу інтуїтивний інтерфейс, можливість редагування конфігурації та інформаційне вікно, що гордо розповідає про витоки програми.

Третій розділ став мостом до користувача, довівши, що конвертер доступний навіть для базових систем і простий у освоєнні. Його інтерфейс, немов добре налаштований інструмент, дозволяє одним рухом переводити одиниці, налаштовувати точність чи додавати нові величини, а детальна інструкція робить цей процес зрозумілим для кожного. Програма готова слугувати як у навчанні, так і в професійній діяльності, стаючи надійним помічником там, де числа потребують ясності.

"Конвертер фізичних величин" досяг поставленої мети – він не лише виконує конвертацію, а й відкриває можливості для вдосконалення, адаптації та творчості. Ця робота стала не просто технічним проєктом, а прикладом того, як технології можуть спрощувати складне, роблячи його доступним. У майбутньому програму можна розширити, додавши підтримку складених одиниць, локалізацію чи інтеграцію з іншими системами, але вже зараз вона є завершеним продуктом, готовим до використання та гідним уваги. Цей шлях від задуму до реалізації підтвердив, що навіть у світі чисел є місце для натхнення й практичної користі.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Глинка Н. Л. Загальна хімія: підручник для вищих навчальних закладів. 6-те вид., перероб. і допов. Київ: Вища школа, 2009. 656 с.
2. Детлаф А. А., Яворський Б. М. Курс фізики: підручник для студентів вищих технічних навчальних закладів. 2-ге вид., перероб. Київ: Вища школа, 2001. 718 с.
3. Кучерук І. М., Горбачук І. Т. Загальний курс фізики: у 3 т. Т. 1. Механіка. Київ: Техніка, 2006. 512 с.
4. Савельєв І. В. Курс загальної фізики: у 3 т. Т. 2. Електрика і магнетизм. Київ: Либідь, 2003. 480 с.
5. Трофимова Т. І. Курс фізики: підручник для студентів вищих навчальних закладів. 10-те вид., стер. Москва: Академія, 2010. 560 с.
6. Бойко В. С., Бойко Р. В. Основи програмування на C#: навчальний посібник. Київ: КНЕУ, 2015. 320 с.
7. Ріхтер Дж. CLR via C#: програмування на платформі Microsoft .NET Framework 4.5 мовою C#. 4-те вид. Київ: ДіаСофт, 2013. 896 с.
8. Троелсен Е., Джепікс Ф. Мова програмування C# 10 і платформа .NET 8. Київ: Вільямс, 2024. 1248 с.
9. Альбахарі Дж., Альбахарі Б. C# 8 у стислому викладі. Київ: Діалектика, 2020. 1056 с.
10. Фленов М. Є. Програмування на C# для чайників. 2-ге вид. Харків: Фоліо, 2018. 352 с.
11. Шилдт Г. C# 4.0: повне керівництво. Київ: Вільямс, 2011. 1056 с.
12. Мак-Дональд М. Windows Forms: програмування для .NET Framework. Київ: ДіаСофт, 2007. 672 с.
13. Петцольд Ч. Програмування для Windows Forms. Київ: Вільямс, 2008. 912 с.
14. Ліберти Дж. Програмування на C# для професіоналів. Київ: Діалектика, 2016. 864 с.

15. Коваленко О. В. Основи роботи з JSON у програмуванні: навчальний посібник. Львів: ЛНУ ім. Івана Франка, 2019. 240 с.
16. Стахів П. Г., Гамола О. І. Фізика: довідник для студентів. Львів: Видавництво Львівської політехніки, 2012. 384 с.
17. Білецький В. С., Олійник Т. А. Основи інформатики та програмування: підручник. Київ: КНУ ім. Тараса Шевченка, 2014. 456 с.
18. Григор'єв В. М. Розробка програмного забезпечення: методичні вказівки до виконання дипломних робіт. Харків: ХНУРЕ, 2020. 128 с.
19. Microsoft .NET 8.0 Documentation [Електронний ресурс] / Microsoft Corporation. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8. – Дата звернення: 04.04.2025.
20. Newtonsoft.Json Documentation [Електронний ресурс] / James Newton-King. – Режим доступу: https://www.newtonsoft.com/json/help/html/Introduction.htm. – Дата звернення: 04.04.2025.
21. Сидоров М. О. Методика розробки графічних інтерфейсів користувача у Windows Forms // Вісник НТУУ "КПІ". Серія: Інформатика, управління та обчислювальна техніка. 2018. № 68. С. 45–52.
22. Козловський О. В. Використання JSON для зберігання конфігураційних даних у програмуванні // Наукові записки УжНУ. Серія: Інформатика. 2021. Вип. 15. С. 23–29.
23. Unit Conversion Reference [Електронний ресурс] / National Institute of Standards and Technology (NIST). – Режим доступу: https://www.nist.gov/pml/special-publication-811. – Дата звернення: 04.05.2025.
24. Фізика в таблицях і формулах: довідник [Електронний ресурс] / Освітній портал "Фізика". – Режим доступу: https://physics.info/conversion/. – Дата звернення: 04.05.2025.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace PhysicalUnitsConverter
{
   partial class Form1
   {
      /// <summary>
      ///  Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      ///  Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      ///  Required method for Designer support - do not modify
```