Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ПЕРЕТВОРЕННЯ МАТЕМАТИЧНИХ КОНСТРУКЦІЙ З ТЕКСТОВИХ ФАЙЛІВ У КОДОВЕ ПОДАННЯ
DESIGN OF A SOFTWARE SYSTEM FOR MATHEMATICAL CONSTRUCT-TO-CODE CONVERSION FROM TEXT FILES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-212
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

ПОЧТАРЬ Б.К.
(ПРІЗВИЩЕ та ініціали)
Керівник СУББОТІН С.О.
(ПРІЗВИЩЕ та ініціали)
Рецензент КОРОТКИЙ О.В.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПОЧТАРЯ Богдана Костянтиновича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання. Design of a Software System for Mathematical Construct-to-Code Conversion from Text Files

керівник проєкту (роботи) д.т.н., професор, СУББОТІН Сергій Олександрович ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від "07" квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 10 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програмної системи. 4. Експлуатація, тестування та експериментальне дослідження програмної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУБОТІН С.О., завідувач кафедри		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 20 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програмної системи.	2 тиждень	Розділ 3
5	Розробка програмної системи.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмної системи.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Богдан ПОЧТАРЬ
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Сергій СУББОТІН
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 91 с., 4 табл., 24 рис., 3 дод., 17 джерел.

C#, VISUAL STUDIO, КОДОВЕ ПОДАННЯ, МОВА ПРОГРАМУВАННЯ, ОБРОБКА ДАНИХ, ПРОГРАМНА СИСТЕМА.

Об’єкт дослідження – процеси, пов’язані зі створенням програмних систем для перетворення даних.

Предмет дослідження – програмні системи для конвертації даних.

Мета роботи – розробка програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання для зменшення витрат людських та часових ресурсів на обробку математичних даних.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Створено програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання. Проведено дослідження розробленої програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання.

Висновки. Мету роботи досягнуто. Розроблено програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – обробка даних.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 91 pages, 4 tables, 24 figures, 3 appendixes, 17 sources.

C#, VISUAL STUDIO, CODE REPRESENTATION, PROGRAMMING LANGUAGE, DATA PROCESSING, SOFTWARE SYSTEM.

The object of research is processes related to the creation of software systems for data conversion.

The subject of the research is software systems for data conversion.

The purpose of this work is to develop the software system for converting mathematical structures from text files into code representation in order to reduce the cost of human and time resources for processing mathematical data.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software system for converting mathematical structures from text files into code representation has been designed. The software system for converting mathematical structures from text files into code representation has been developed. The study of the developed software system for converting mathematical structures from text files into code representation has been conducted.

Conclusions. The goal of the work has been achieved. The software system for converting mathematical structures from text files into code representation using the C# programming language and the Visual Studio development environment has been developed.

Scope of use – data processing.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

КП – кодове подання;

МК – математична конструкція;

ПС – програмна система.

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій та зростання обсягів науково-технічної інформації дедалі більшого значення набуває автоматизація процесів обробки формалізованих знань. Значна частина таких знань подається у вигляді математичних конструкцій, які містяться у текстових документах, наукових публікаціях, технічній документації та навчальних матеріалах. Водночас їх використання у програмному середовищі часто ускладнюється необхідністю ручного перенесення, інтерпретації та адаптації до конкретних мов програмування або систем обчислень. Це створює додаткові витрати часу, підвищує ймовірність помилок та знижує ефективність роботи спеціалістів [1]-[2].

З огляду на це, підвищується потреба у створенні програмної системи, здатної здійснювати перетворення математичних конструкцій із текстового подання у формалізований код, придатний для подальшого використання в інформаційних системах. Актуальність зазначеного напряму також обумовлюється розвитком штучного інтелекту та систем обробки природної мови, які відкривають нові можливості для аналізу складних текстових структур. Використання таких технологій у контексті розпізнавання та інтерпретації математичних виразів дозволяє значно розширити функціональність програмних систем і підвищити рівень їх інтелектуалізації. У результаті створюються передумови для інтеграції різнорідних інформаційних ресурсів та побудови єдиних середовищ підтримки наукових досліджень і інженерної діяльності [3]-[4].

Таким чином, необхідність створення програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання визначається сукупністю факторів, серед яких ключовими є зростання обсягів інформації, ускладнення математичних моделей, потреба в автоматизації та інтеграції знань, а також прагнення до підвищення ефективності наукової, освітньої та інженерної діяльності [2]-[5].

Проте деякі програмні системи перетворення даних орієнтовані на конкретні формати вхідних даних або певні математичні нотації. У разі відхилення від передбачених шаблонів ефективність обробки істотно зменшується, а інколи процес перетворення стає неможливим без попередньої ручної підготовки тексту. Це частково нівелює переваги автоматизації та вимагає додаткових ресурсів. Проблеми виникають також на етапі семантичного аналізу, оскільки математичні конструкції часто мають неоднозначну інтерпретацію без урахування контексту. Крім того, розробка та підтримка таких програмних рішень вимагає залучення фахівців високого рівня, здатних поєднувати знання у галузях математики, програмування та обробки природної мови. Це ускладнює масштабування систем і підвищує вартість їх впровадження та супроводу [3]-[5].

Тому актуальною є розробка програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. У дипломній кваліфікаційній роботі бакалавра розв’язується актуальне завдання розробки програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання для зменшення витрат людських та часових ресурсів на обробку математичних даних.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв’язати такі задачі:

– виконати аналіз предметної області та програмних систем для перетворення даних;

– здійснити проєктування програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання;

– створити програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання;

– дослідити розроблену програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмні системи для перетворення даних

Програмна система перетворення математичних конструкцій з текстових файлів у кодове подання являє собою спеціалізований програмний інструмент, призначений для автоматизованого переходу від текстового опису математичних виразів до їх формалізованого представлення у вигляді програмного коду. У межах такого підходу забезпечується інтерпретація математичних записів, їх структуризація та подальше відтворення у вигляді синтаксично коректних конструкцій певної мови програмування. Основною метою функціонування подібних систем виступає спрощення роботи з математичними даними та підвищення ефективності їх використання у програмному середовищі [1]-[2].

У процесі роботи таких програмних систем відбувається послідовна обробка вхідної інформації, яка включає аналіз тексту, виділення окремих елементів математичного виразу та встановлення логічних зв’язків між ними. На основі цього формується внутрішня модель, що відображає структуру та зміст математичної конструкції. Надалі ця модель використовується для генерації програмного коду, який відповідає правилам обраної мови програмування та може бути безпосередньо використаний у розробці програмного забезпечення [1]-[2].

Особливістю таких програмних систем є орієнтація на точність відтворення математичних залежностей. Забезпечується коректне врахування порядку виконання операцій, вкладеності виразів і специфіки математичних функцій. Це дозволяє зберігати змістовну відповідність між початковим записом і отриманим кодом. Водночас важливим аспектом є використання проміжного структурованого представлення, яке дозволяє відокремити етап аналізу від етапу генерації результату та забезпечує гнучкість у виборі формату вихідних даних [1]-[2].

До переваг таких систем відноситься зменшення обсягу ручної роботи під час перенесення математичних формул у програмний код, що сприяє підвищенню продуктивності та зниженню ймовірності виникнення помилок. Автоматизація цього процесу дозволяє уникнути неточностей, які можуть виникати при ручному введенні складних виразів, а також забезпечує уніфікацію підходів до їх представлення. Крім того, такі системи сприяють покращенню взаємодії між різними програмними компонентами, оскільки використовують стандартизовані формати даних [1]-[2].

Програмні системи перетворення математичних конструкцій у кодове подання виступають важливим засобом автоматизації обробки формалізованої інформації, поєднуючи можливості аналізу, структуризації та генерації програмних рішень. Їх застосування дозволяє оптимізувати процеси розробки та забезпечити більш ефективне використання математичних моделей у програмному забезпеченні [1]-[2].

Процес розробки програмних систем перетворення математичних конструкцій з текстових файлів у кодове подання розглядається як складний і багатоетапний цикл створення програмного продукту, у межах якого поєднуються підходи з аналізу текстових даних, формалізації математичних знань і генерації програмного коду. На початкових етапах формується загальне уявлення про призначення системи та визначаються вимоги до її функціонування, що дозволяє окреслити межі застосування та основні сценарії використання. Особлива увага при розробці таких програмних систем приділяється характеристикам вхідних даних, оскільки саме вони визначають складність майбутніх алгоритмів обробки [3]-[4].

У подальшому здійснюється проєктування загальної архітектури системи, в межах якої визначається структура компонентів та принципи їх взаємодії. Важливим аспектом цього етапу є розподіл функціональності між модулями, що відповідають за аналіз тексту, побудову внутрішнього представлення математичних конструкцій і генерацію програмного коду. Формується концепція проміжного формату даних, який забезпечує

узгодженість між різними етапами обробки та дозволяє зберігати логіку математичних виразів у структурованому вигляді [3]-[4].

Подальша розробка пов’язується зі створенням механізмів синтаксичного та семантичного аналізу, які дозволяють інтерпретувати текстові математичні конструкції. У цьому контексті реалізуються алгоритми розбору, що визначають структуру виразів, встановлюють зв’язки між їх елементами та формують внутрішню модель, придатну для подальшої обробки. Особливу складність становить забезпечення коректності такого аналізу, оскільки необхідно враховувати різноманіття способів запису математичних формул [3]-[4].

Наступним етапом стає реалізація механізмів генерації програмного коду, які трансформують сформовану модель у синтаксично правильні конструкції обраної мови програмування. При цьому важливо забезпечити відповідність між математичним змістом виразу та його програмною інтерпретацією, що досягається шляхом коректного відображення операцій, функцій і порядку їх виконання. Паралельно вирішуються питання оптимізації сформованого коду та його адаптації до вимог конкретного середовища виконання [3]-[4].

Значна увага приділяється розробці інтерфейсу користувача, який забезпечує взаємодію із системою та надає засоби введення, перегляду та збереження результатів. Інтерфейс має бути зрозумілим і логічно організованим, щоб користувач міг ефективно керувати процесом перетворення без необхідності глибокого занурення у внутрішні механізми системи [4]-[5].

Після реалізації основної функціональності проводиться тестування, спрямоване на перевірку правильності роботи всіх компонентів системи. Особлива увага приділяється коректності обробки різних типів математичних виразів, включаючи складні та нестандартні конструкції. У ході тестування виявляються можливі помилки та неточності, які усуваються для забезпечення стабільності та надійності роботи програмного продукту [4]-[5].

Завершальні етапи розробки пов’язані з оптимізацією системи, її документуванням та підготовкою до використання у реальних умовах. Передбачається можливість подальшого розширення функціональності та адаптації до нових вимог, що робить систему гнучкою та придатною до розвитку [4]-[5].

Таким чином, процес створення таких програмних систем характеризується поєднанням аналітичних, алгоритмічних і інженерних підходів, що дозволяє забезпечити ефективне перетворення математичних конструкцій у програмне подання та створити інструмент, здатний автоматизувати складні операції обробки формалізованих даних [4]-[5].

У контексті дослідження програмних систем перетворення математичних конструкцій з текстових файлів у кодове подання доцільно додатково розглянути їх як інструменти, що поєднують формальні методи опису знань із практичними засобами автоматизації обчислювальних процесів. Такі системи виступають своєрідним містком між людським способом подання математичної інформації та машинною інтерпретацією, що дозволяє використовувати формули не лише як елемент опису, але й як безпосередній компонент програмної логіки [4]-[5].

Функціонування подібних програмних систем тісно пов’язане з проблемою однозначності інтерпретації математичних записів. У природному вигляді математичні конструкції можуть допускати варіативність запису, що ускладнює їх автоматичне опрацювання. У зв’язку з цим особливого значення набувають механізми нормалізації та стандартизації вхідних даних, які забезпечують приведення різних варіантів запису до уніфікованого формату. Це, у свою чергу, створює передумови для підвищення точності та передбачуваності результатів перетворення [4]-[5].

У процесі розширення функціональності виникає необхідність підтримки нових типів математичних операцій, функцій і структур, що вимагає гнучкої архітектури та можливості модифікації без істотного впливу на вже реалізовані компоненти. У цьому контексті важливим стає

використання модульного підходу, який дозволяє поступово розширювати можливості системи відповідно до нових вимог [4]-[5].

Математичні конструкції широко застосовуються у фізиці, інженерії, економіці та інших галузях, кожна з яких має власні особливості запису та інтерпретації формул. Відповідно, програмні системи повинні враховувати ці специфічні вимоги, що потребує розширення словників символів, підтримки спеціалізованих функцій та врахування контексту використання [4]-[5].

Не менш важливим є аспект взаємодії з іншими інформаційними системами. Програмні засоби перетворення математичних конструкцій можуть виступати як складова частина більших програмних комплексів, забезпечуючи передачу даних між різними модулями та форматами. У цьому випадку ключову роль відіграє підтримка стандартів обміну даними, що дозволяє інтегрувати результати перетворення у ширші процеси обробки інформації [4]-[5].

Таким чином, програмні системи цього типу часто є не лише інструментами для виконання окремої функції, але і складовими елементами більш загальної екосистеми обробки формалізованих знань. Їх розвиток сприяє підвищенню рівня автоматизації, покращенню якості обробки математичних даних і розширенню можливостей використання формул у програмному середовищі [4]-[5].

Визначено, що програмна система перетворення математичних конструкцій з текстових файлів у кодове подання являє собою спеціалізований програмний інструмент, призначений для автоматизованого переходу від текстового опису математичних виразів до їх формалізованого представлення у вигляді програмного коду. У межах такого підходу забезпечується інтерпретація математичних записів, їх структуризація та подальше відтворення у вигляді синтаксично коректних конструкцій певної мови програмування. Основною метою функціонування подібних систем виступає спрощення роботи з математичними даними та підвищення ефективності їх використання у програмному середовищі.

1.2 Аналіз програмних систем для перетворення даних

Проаналізуємо програмні системи для перетворення даних [6]-[11].

Програмна система Mathpix (рис. 1.1) є інструментом для автоматизованого розпізнавання та перетворення математичних конструкцій у структуровані формати, придатні для подальшого використання у програмному середовищі або текстових редакторах. Її призначення полягає у спрощенні процесу роботи з математичними виразами шляхом переведення їх із неформалізованого або візуального представлення у стандартизоване кодове подання. Такий підхід дозволяє значно скоротити час, необхідний для ручного введення складних формул, а також зменшити кількість помилок, пов'язаних із людським фактором [6].

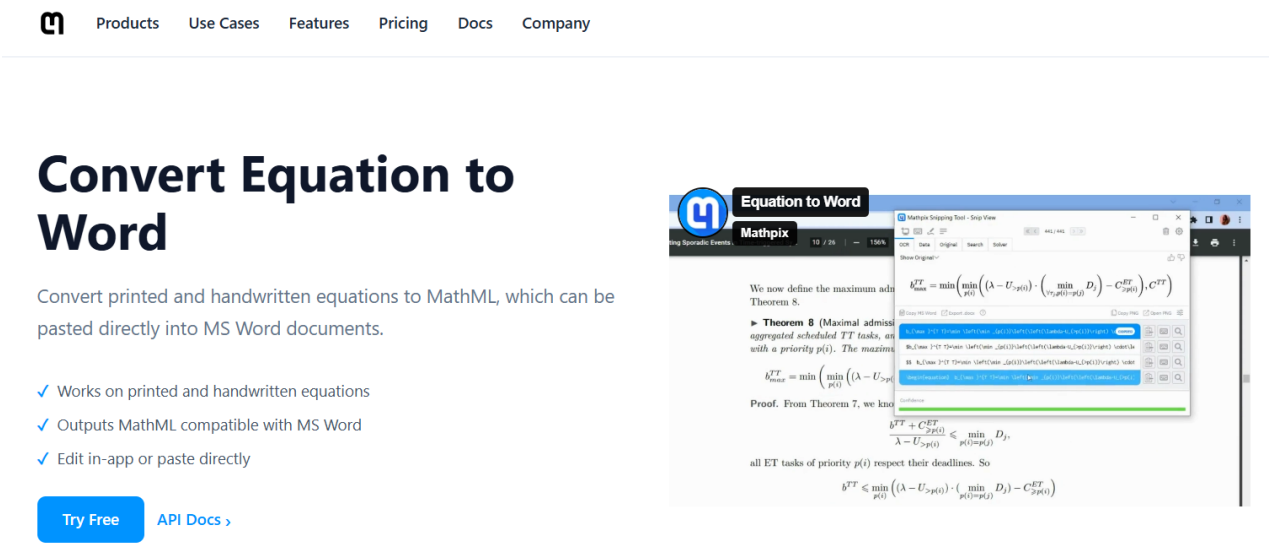


Рисунок 1.1 – Програмна система Mathpix [6]

Функціональні можливості системи Mathpix ґрунтуються на поєднанні технологій розпізнавання зображень, аналізу структури математичних виразів і генерації відповідного текстового або кодового представлення. Вхідні дані можуть бути представлені у вигляді тексту або зображення, після чого відбувається їх інтерпретація з урахуванням синтаксичних і семантичних особливостей математичної нотації. Результатом є формалізований запис, який

може бути використаний у різних форматах, включаючи LaTeX, MathML або інші варіанти, сумісні з програмними системами [6].

Особливістю програмного продукту Mathpix є здатність коректно обробляти складні математичні структури, що містять вкладені вирази, функції, індекси та спеціальні символи. Забезпечується точне відтворення логіки математичних залежностей, що є критично важливим для подальшого використання отриманого результату у програмному коді або наукових документах. Крім того, система орієнтована на інтеграцію з популярними текстовими редакторами, що дозволяє здійснювати перетворення безпосередньо у процесі роботи з документами [6].

З точки зору програмної реалізації, функціонування системи Mathpix базується на використанні хмарної архітектури, що забезпечує обробку даних на віддалених серверах. Такий підхід дозволяє застосовувати складні алгоритми аналізу без необхідності встановлення потужного програмного забезпечення на локальному пристрої. Взаємодія з користувачем реалізується через веб-інтерфейс або клієнтські застосунки, які забезпечують зручний доступ до основних функцій системи. Обробка математичних конструкцій включає етапи попередньої підготовки даних, розпізнавання, побудови внутрішньої структури та генерації результату у заданому форматі [6].

Важливою характеристикою програмної системи Mathpix є адаптивність системи до різних стилів запису математичних виразів, що досягається завдяки використанню інтелектуальних алгоритмів аналізу. Це дозволяє забезпечити високу точність розпізнавання навіть у випадках, коли вихідні дані мають складну або нетипову структуру. У результаті формується універсальний інструмент, який може застосовуватися у навчальній, науковій та прикладній діяльності [6].

Таким чином, програмна система Mathpix являє собою ефективне рішення для автоматизованого перетворення математичних конструкцій у кодове подання, поєднуючи сучасні технології розпізнавання та структурованого аналізу з можливостями інтеграції у різні програмні

середовища. Це забезпечує високий рівень зручності використання та відкриває широкі перспективи для автоматизації роботи з математичними даними [6].

Основними характеристиками програмної системи Mathpix є такі [6]:

– автоматизоване розпізнавання формул: у програмній системі Mathpix забезпечується перетворення математичних конструкцій із текстового або графічного представлення у структурований формат без необхідності ручного введення [6];

– підтримка різних форматів виводу: результати роботи програмної системи Mathpix можуть бути представлені у вигляді LaTeX, MathML або інших форматів, придатних для подальшого використання у програмних системах [6];

– інтеграція з редакторами: у програмній системі Mathpix реалізується можливість використання функціоналу безпосередньо у текстових середовищах, що спрощує роботу з документами [6];

– висока точність обробки: у програмній системі Mathpix забезпечується коректне відтворення складних математичних виразів із урахуванням вкладеності та пріоритетів операцій [6];

– підтримка складних структур: у програмній системі Mathpix передбачається робота з функціями, індексами, дробами та іншими елементами математичної нотації [6];

– використання хмарної обробки: основні обчислення виконуються на віддалених серверах, що знижує вимоги до локального обладнання [6];

– зручний інтерфейс користувача: у програмній системі Mathpix забезпечується інтуїтивна взаємодія з системою та швидкий доступ до основних функцій [6];

– адаптивність до різних записів: програмна система Mathpix здатна працювати з різними стилями представлення математичних конструкцій [6];

– швидкість обробки даних: забезпечується оперативне перетворення навіть складних математичних виразів [6];

– можливість використання у різних сферах: програмна система Mathpix застосовується у навчанні, наукових дослідженнях та інженерній діяльності [6].

Серед недоліків програмної системи Mathpix можна відзначити залежність від мережевого підключення, оскільки основна обробка виконується у хмарному середовищі, а також обмеження безкоштовного використання, що може ускладнювати застосування у тривалих або ресурсомістких задачах. Крім того, у деяких випадках можливі неточності під час розпізнавання нестандартних або складно структурованих математичних виразів, що потребує додаткової перевірки отриманих результатів [6].

Програмна система Bibcit (рис. 1.2) орієнтована на перетворення математичних конструкцій у формалізоване представлення, придатне для використання у текстових редакторах та програмних середовищах [7].

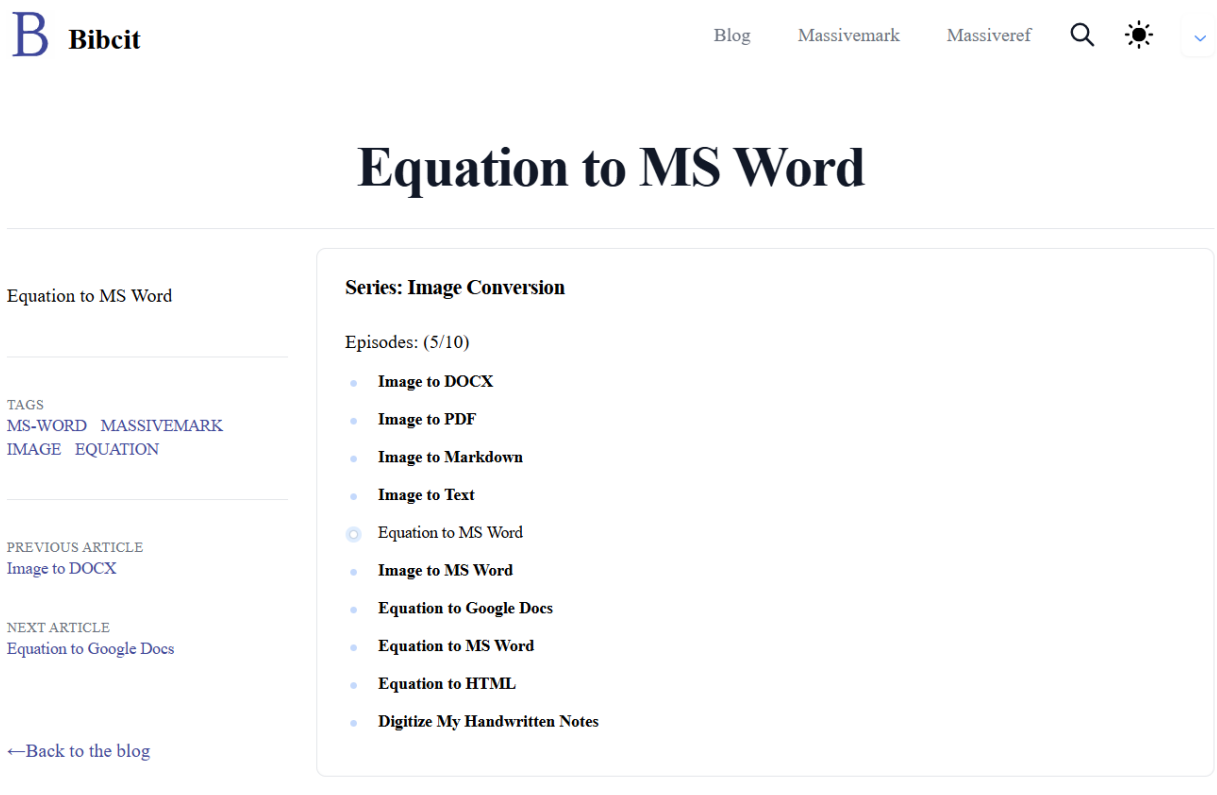


Рисунок 1.2 – Програмна система Bibcit [7]

Призначення програмної системи Bibcit пов’язується із автоматизацією процесу перенесення математичних виразів із неструктурованого або напівструктурованого вигляду у формат, який може бути інтерпретований програмними засобами або використаний у подальшій обробці. Такий підхід дозволяє зменшити складність роботи з математичними формулами та підвищити ефективність підготовки документів і програмного коду [7].

Функціонування програмної системи Bibcit базується на обробці вхідних даних, які можуть бути представлені у вигляді тексту або зображення математичних виразів, з подальшим їх аналізом і трансформацією у внутрішнє структуроване представлення. У процесі обробки виконується ідентифікація елементів формули, визначення їх взаємозв’язків та формування узгодженої моделі, що відображає логіку математичної конструкції. Результатом такої обробки стає представлення, сумісне із середовищами редагування документів, зокрема з можливістю подальшого використання у вигляді форматованих формул або кодових фрагментів [7].

Особливістю програмної системи Bibcit є орієнтація на інтеграцію з текстовими процесорами, що дозволяє безпосередньо переносити результати перетворення у документи без необхідності додаткового редагування. Забезпечується узгодженість між вхідним математичним записом та його відображенням у середовищі редагування, що сприяє збереженню змістовної точності та структурної цілісності формул. У межах функціонування системи враховуються особливості математичної нотації, включаючи вкладені вирази, спеціальні символи та функціональні залежності [7].

З точки зору програмної реалізації, програмна система Bibcit використовує підхід, що поєднує алгоритми розпізнавання, структурного аналізу та генерації результату у заданому форматі. Передбачається застосування веб-орієнтованої архітектури, у межах якої обробка даних може виконуватися на стороні сервера, що дозволяє використовувати складні алгоритми без значного навантаження на клієнтський пристрій. Інтерфейс

взаємодії забезпечується через вебсередовище, що робить доступ до функціональності незалежним від конкретної платформи [7].

Важливою характеристикою програмної системи Bibcit є здатність системи адаптуватися до різних варіантів запису математичних конструкцій, що досягається завдяки використанню механізмів узагальненого аналізу та нормалізації вхідних даних. Це дозволяє підвищити точність перетворення та забезпечити коректне відображення навіть складних виразів. У результаті формується універсальний інструмент, який може бути використаний як у навчальній діяльності, так і в професійних задачах, пов’язаних із обробкою математичної інформації [7].

Таким чином, програмна система Bibcit постає як засіб автоматизованого перетворення математичних конструкцій у структуроване подання, що поєднує можливості аналізу, інтерпретації та інтеграції з програмними середовищами, забезпечуючи зручність використання та підвищення продуктивності роботи з математичними даними [7].

Програмна система Bibcit має такі особливості [7]:

– автоматизоване перетворення формул: програмна система Bibcit забезпечує трансформацію математичних конструкцій у формат, придатний для використання у текстових редакторах та програмному коді [7];

– підтримка роботи з зображеннями: у програмній системі Bibcit передбачається можливість обробки математичних виразів, представлених у вигляді графічних об’єктів [7];

– інтеграція з текстовими редакторами: у програмній системі Bibcit забезпечується безпосереднє використання результатів у середовищах редагування документів без додаткового форматування [7];

– структурний аналіз виразів: у програмній системі Bibcit виконується розбір математичних конструкцій із визначенням їх ієрархії та логічних зв’язків [7];

– зручність використання: у програмній системі Bibcit забезпечується доступ до функціональності через простий і зрозумілий інтерфейс [7];

– підтримка стандартних форматів: у програмній системі Bibcit результати можуть бути представлені у форматах, сумісних із поширеними системами обробки математичних даних [7];

– веборієнтована архітектура: обробка даних у програмній системі Bibcit здійснюється через мережеві сервіси без необхідності складного локального встановлення [7];

– обробка складних математичних структур: підтримується робота з вкладеними виразами, функціями та спеціальними символами [7];

– адаптивність до різних форматів запису: система здатна інтерпретувати різні стилі представлення математичних формул [7];

– швидке отримання результату: у програмній системі Bibcit забезпечується оперативне формування вихідного представлення без значних затримок [7].

Серед недоліків програмної системи Bibcit можна відзначити залежність від якості вхідних даних, оскільки неточності у вихідному зображенні або тексті можуть призводити до помилок у результаті перетворення. Також обмеження функціональності у безкоштовному доступі може ускладнювати використання системи у розширених сценаріях. Додатково варто враховувати, що у складних випадках із нестандартними математичними записами може виникати необхідність ручного коригування отриманого результату [7].

Програмна система MathML editor (рис. 1.3) є веборієнтованим інструментом, призначеним для створення, редагування та перетворення математичних конструкцій у формалізоване подання, придатне для використання у програмному коді та цифрових документах. Основне її призначення пов’язується із забезпеченням зручного переходу від неструктурованого або напівструктурованого математичного запису до стандартизованого формату, зокрема MathML або LaTeX, що дозволяє інтегрувати отримані результати у різні програмні середовища та інформаційні системи [8].

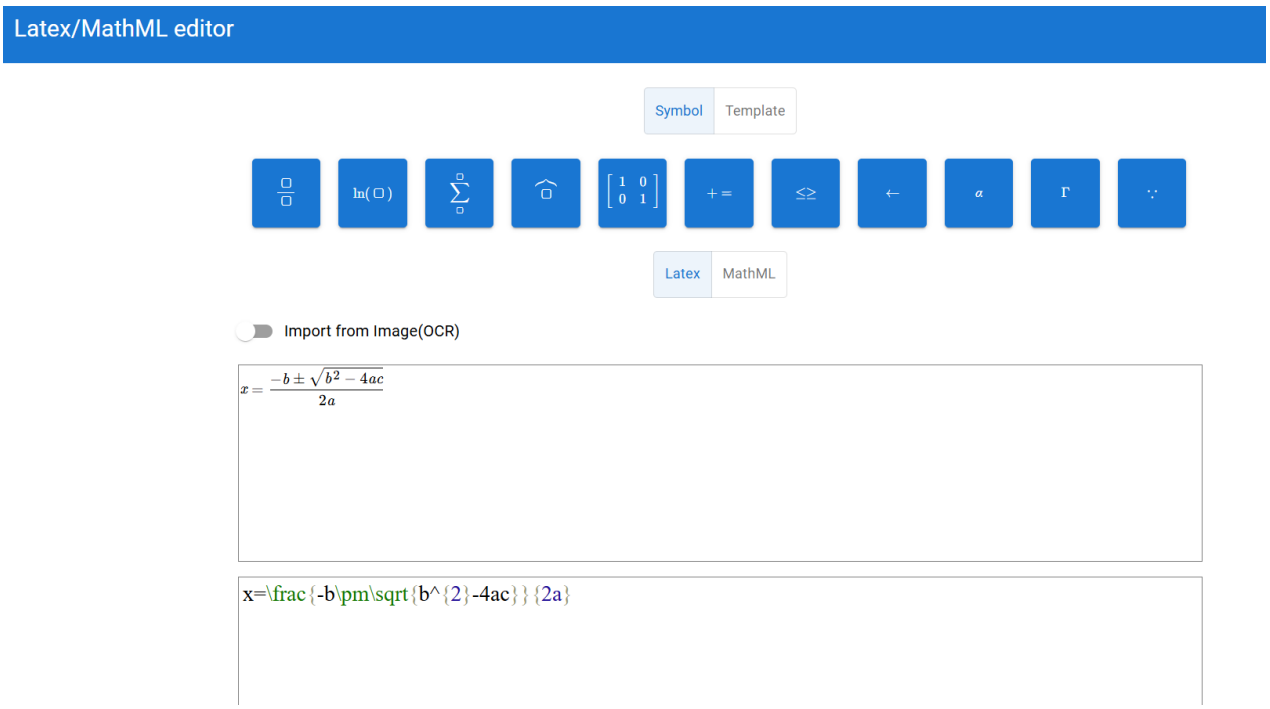


Рисунок 1.3 – Програмна система MathML editor [8]

Функціонування програмної системи MathML базується на поєднанні візуального редагування та текстового представлення математичних виразів. Забезпечується можливість формування формул у графічному режимі, що дозволяє створювати складні математичні конструкції без необхідності безпосереднього написання коду. Одночасно підтримується режим роботи з текстовим поданням, у якому користувач може редагувати вирази у вигляді формалізованої нотації. Така двоїста модель взаємодії забезпечує гнучкість використання та адаптацію до різних рівнів підготовки користувачів [8].

Особливістю програмної системи MathML є можливість перетворення математичних виразів, отриманих із зображень або інших джерел, у структуроване представлення. Використання механізмів розпізнавання дозволяє інтерпретувати математичні конструкції та переводити їх у формат, який може бути оброблений програмними засобами. Це значно спрощує роботу з уже існуючими матеріалами та сприяє автоматизації процесу створення цифрового контенту [8].

З точки зору програмної реалізації, система MathML функціонує як клієнт-серверний вебзастосунок, у якому основна взаємодія здійснюється

через браузер. Інтерфейс реалізується із використанням сучасних веб-технологій, що забезпечують динамічне оновлення вмісту та інтерактивну роботу з формулами. Обробка математичних конструкцій може включати використання бібліотек для рендерингу та інтерпретації, що підтримують різні формати представлення математичних даних. У цьому контексті MathML виступає як стандарт, заснований на XML, який дозволяє описувати як структуру, так і зміст математичних виразів, що є важливим для їх подальшого машинного опрацювання [8].

Важливою характеристикою MathML є підтримка режиму миттєвого відображення результату, коли зміни у формулі одразу відображаються у відповідному представленні. Це дозволяє контролювати процес створення математичних конструкцій у реальному часі та забезпечує високий рівень зручності використання. Додатково передбачається можливість експорту результатів у різні формати, що сприяє їх подальшому використанню у програмному коді або документах [8].

Таким чином, програмна система MathML Editor являє собою універсальний інструмент для створення та перетворення математичних конструкцій, який поєднує візуальні засоби редагування, підтримку формалізованих мов опису та можливості інтеграції з іншими програмними середовищами. Це дозволяє ефективно вирішувати задачі автоматизації обробки математичних даних та підвищує зручність роботи з формулами у цифровому середовищі [8].

Програмна система MathML editor має такі особливості [8]:

– візуальне редагування формул: програмна система MathML editor забезпечує створення математичних конструкцій у графічному режимі без необхідності ручного написання коду [8];

– підтримка MathML: у програмній системі MathML editor реалізується формування структурованого подання математичних виразів у форматі, орієнтованому на машинну обробку [8];

– інтерактивна зміна виразів: у програмній системі MathML editor передбачається миттєве оновлення відображення формули під час її редагування [8];

– робота у браузері: доступ до функціональності здійснюється без встановлення додаткового програмного забезпечення [8];

– підтримка LaTeX: у програмній системі MathML editor забезпечується можливість використання альтернативного текстового представлення математичних конструкцій [8];

– зручний інтерфейс: у програмній системі MathML editor реалізується інтуїтивно зрозуміле середовище для роботи з математичними виразами [8];

– структуроване представлення даних: у програмній системі MathML editor забезпечується побудова ієрархічної моделі математичних конструкцій [8];

– можливість експорту результатів: у програмній системі MathML editor передбачається збереження сформованих виразів для подальшого використання [8];

– підтримка складних виразів: у програмній системі MathML editor забезпечується робота з вкладеними конструкціями, функціями та спеціальними символами [8];

– інтеграційна придатність: результати програмної системи MathML editor можуть бути використані у різних програмних середовищах та документах [8].

Серед недоліків програмної системи MathML editor можна відзначити відсутність розвинених засобів автоматичного розпізнавання математичних конструкцій із зображень, що обмежує можливості автоматизації. Також певні труднощі можуть виникати при роботі зі складними або нетиповими виразами, які потребують більш гнучких механізмів налаштування. Крім того, залежність від вебсередовища може створювати обмеження у випадках відсутності стабільного доступу до мережі [8].

Існують й інші системи для конвертації математичних виразів [9]-[11].

Порівняльну характеристику програмних систем для перетворення даних наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмних систем для перетворення даних

Критерій порівняння	Mathpix [6]	Bibcit [7]	MathML editor [8]
Робота з текстовими файлами	+	+–	+
Підтримка складних математичних конструкцій	+	+–	+–
Інтеграція з текстовими редакторами	+–	+	+–
Редагування формул у реальному часі	–	–	+
Хмарна обробка	+	+	+–
Гнучкість налаштування формату виводу	+–	+	+–

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних систем для перетворення даних. Проте деякі програмні системи перетворення даних орієнтовані на конкретні формати вхідних даних або певні математичні нотації. У разі відхилення від передбачених шаблонів ефективність обробки істотно зменшується, а інколи процес перетворення стає неможливим без попередньої ручної підготовки тексту. Це частково нівелює переваги автоматизації та вимагає додаткових ресурсів. Проблеми виникають також на етапі семантичного аналізу, оскільки математичні конструкції часто мають неоднозначну інтерпретацію без урахування контексту. Крім того, розробка та підтримка таких програмних рішень вимагає залучення фахівців високого рівня, здатних поєднувати знання у галузях математики, програмування та обробки природної мови. Це

ускладнює масштабування систем і підвищує вартість їх впровадження та супроводу. Тому актуальною є розробка програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання.

При розробці програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання необхідно забезпечити такі функціональні вимоги:

– підтримка можливості зчитування математичних конструкцій із текстових файлів та їх попереднього опрацювання для подальшого аналізу;

– підтримка можливості синтаксичного розбору математичних виразів із урахуванням операторів, функцій, змінних та вкладених структур;

– підтримка можливості формування внутрішнього представлення математичних конструкцій на основі XML-подібної моделі;

– підтримка можливості обробки спеціальних символів, зокрема літер грецького алфавіту та інших математичних позначень;

– можливість генерації програмного коду на основі структурованого представлення математичних виразів;

– підтримка можливості коректного врахування пріоритетів операцій та вкладеності виразів під час генерації коду;

– можливість відображення результатів перетворення у зручному для користувача вигляді в інтерфейсі програми.

1.3 Висновки за розділом 1

Визначено, що програмна система перетворення математичних конструкцій з текстових файлів у кодове подання являє собою спеціалізований програмний інструмент, призначений для автоматизованого переходу від текстового опису математичних виразів до їх формалізованого представлення у вигляді програмного коду. У межах такого підходу забезпечується інтерпретація математичних записів, їх структуризація та подальше відтворення у вигляді синтаксично коректних конструкцій певної мови

програмування. Основною метою функціонування подібних систем виступає спрощення роботи з математичними даними та підвищення ефективності їх використання у програмному середовищі.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних систем для перетворення даних. Проте деякі програмні системи перетворення даних орієнтовані на конкретні формати вхідних даних або певні математичні нотації. У разі відхилення від передбачених шаблонів ефективність обробки істотно зменшується, а інколи процес перетворення стає неможливим без попередньої ручної підготовки тексту. Це частково нівелює переваги автоматизації та вимагає додаткових ресурсів. Проблеми виникають також на етапі семантичного аналізу, оскільки математичні конструкції часто мають неоднозначну інтерпретацію без урахування контексту. Крім того, розробка та підтримка таких програмних рішень вимагає залучення фахівців високого рівня, здатних поєднувати знання у галузях математики, програмування та обробки природної мови. Це ускладнює масштабування систем і підвищує вартість їх впровадження та супроводу. Тому актуальною є розробка програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання.

Сформульовано функціональні вимоги до програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання було використано мову програмування C# [12]-[13].

Мова програмування C# є однією з ключових технологій сучасної розробки програмного забезпечення, що поєднує у собі риси строгої формалізації та практичної зручності застосування. Її еволюція відбувалася у тісному зв’язку з розвитком платформи .NET, що зумовило глибоку інтеграцію з інструментами виконання, бібліотеками та середовищами розробки. У результаті сформовано цілісне середовище, у межах якого забезпечується узгодженість між мовними конструкціями, механізмами виконання програм і засобами налагодження [12]-[13].

Характерною рисою C# є поєднання декларативних і імперативних підходів до програмування, що дозволяє описувати як логіку обчислень, так і структуру даних на високому рівні абстракції. Значна увага приділяється читабельності коду та його підтримуваності, що досягається завдяки чітко визначеному синтаксису, передбачуваній поведінці мовних конструкцій і наявності сучасних засобів організації коду. Це створює передумови для розробки складних систем, у яких важливо зберігати прозорість архітектурних рішень [12]-[13].

Важливим аспектом є підтримка багатопарадигмальності, що дозволяє використовувати елементи функціонального програмування поряд із класичним об’єктно-орієнтованим підходом. Така гнучкість сприяє вибору найбільш доцільних методів реалізації залежно від характеру задачі. Зокрема, можливість застосування лямбда-виразів, запитів до колекцій і механізмів відкладеного виконання дозволяє ефективно працювати з потоками даних і складними структурами [12]-[13].

Значну роль відіграє автоматичне керування пам’яттю, яке реалізується через механізм збирання сміття. Це зменшує ризик виникнення помилок, пов’язаних із ручним управлінням ресурсами, і дозволяє зосередитися на логіці прикладної задачі. Водночас забезпечується достатній рівень контролю над продуктивністю, що є важливим для ресурсомістких застосувань [12], [13].

Не можна залишити поза увагою і розвинені можливості паралельного та асинхронного програмування. Використання сучасних мовних конструкцій дає змогу ефективно організовувати обробку великої кількості задач, що виконуються одночасно, без суттєвого ускладнення коду. Це особливо актуально в умовах зростання вимог до швидкодії та масштабованості програмних систем [12]-[13].

Додатково варто відзначити високий рівень стандартизації та стабільності мови, що забезпечує її сумісність між різними версіями та платформами. Це створює сприятливі умови для довготривалого використання розроблених програмних продуктів і спрощує їх модернізацію. У поєднанні з активною підтримкою спільноти та постійним розвитком інструментів це робить мову програмування C# привабливим вибором для широкого спектра задач, починаючи від прикладних систем і завершуючи складними інженерними рішеннями [12], [13].

Отже, мова програмування C# є універсальним інструментом, який забезпечує баланс між строгістю формалізації, гнучкістю підходів і практичною ефективністю, що дозволяє використовувати його у різноманітних напрямах сучасної розробки ПЗ [12]-[13].

Вибір мови програмування для реалізації програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання значною мірою впливає на ефективність, надійність і подальшу підтримку такого рішення. Доцільність використання C# у цьому контексті зумовлюється сукупністю її технічних характеристик і можливостей, які добре відповідають вимогам подібних систем [12]-[13].

Передусім варто врахувати високий рівень розвитку екосистеми платформи .NET, яка забезпечує широкий набір вбудованих засобів для роботи з текстом, файлами та структурами даних. Це дозволяє ефективно реалізовувати складні алгоритми аналізу та обробки текстових джерел, у яких містяться математичні конструкції. Наявність потужних бібліотек для роботи з регулярними виразами, потоками даних і колекціями значно спрощує розробку компонентів, відповідальних за синтаксичний аналіз і трансформацію інформації [12]-[13].

Важливим аргументом є також сувора типізація мови, яка сприяє зменшенню кількості помилок під час реалізації алгоритмів перетворення. У задачах, пов’язаних із математичними обчисленнями та побудовою формалізованих моделей, це має особливе значення, оскільки дозволяє забезпечити коректність операцій і передбачуваність результатів. Механізми обробки винятків і вбудовані засоби діагностики додатково підвищують надійність програмної системи [12]-[13].

Доцільність використання C# також пояснюється підтримкою об’єктно-орієнтованого підходу, який дозволяє природно моделювати математичні конструкції у вигляді ієрархій класів та абстракцій. Це спрощує розширення функціональності системи, наприклад, у разі необхідності додавання нових типів виразів або правил перетворення. Така гнучкість є важливою для систем, що мають працювати з різноманітними математичними структурами [12]-[13].

Окремої уваги заслуговують можливості інтеграції з іншими технологіями та сервісами. Мова програмування C# забезпечує зручну взаємодію з базами даних, вебсервісами та інтерфейсами користувача, що відкриває перспективи для створення комплексних програмних рішень. Це дозволяє не лише виконувати перетворення математичних конструкцій, а й організовувати їх зберігання, візуалізацію та подальше використання у різних прикладних середовищах [12]-[13].

Не менш важливим є високий рівень продуктивності, який досягається завдяки оптимізаціям платформи .NET та використанню сучасних механізмів

керування пам’яттю. Це дає змогу ефективно обробляти великі обсяги текстових даних без істотних втрат швидкодії, що є актуальним для задач інтелектуального аналізу інформації [12]-[13].

Таким чином, застосування мови програмування C# для створення програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання є обґрунтованим завдяки поєднанню розвиненої інфраструктури, надійності, гнучкості та продуктивності. Це створює сприятливі умови для розробки ефективного, масштабованого та зручного у супроводі програмного продукту [12]-[13].

Обґрунтування вибору мови програмування для розробки програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Зручність роботи з текстом	+	+	+–
Строга типізація	+	–	+
Продуктивність	+	+–	+
Простота реалізації складної логіки	+	+	+–
Зручність налагодження	+	+	+–

Отже, для реалізації програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано мову програмування C#, що забезпечує широкий набір вбудованих засобів для роботи з текстом, файлами та структурами даних. Це дозволяє ефективно

реалізовувати складні алгоритми аналізу та обробки текстових джерел, у яких містяться математичні конструкції. Наявність потужних бібліотек для роботи з регулярними виразами, потоками даних і колекціями значно спрощує розробку компонентів, відповідальних за синтаксичний аналіз і трансформацію інформації.

2.2 Вибір середовища розробки для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

В якості середовища розробки для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання було обрано середовище Visual Studio [14]-[15].

Середовище розробки Visual Studio характеризується як багатофункціональна платформа, що забезпечує не лише написання програмного коду, але й комплексну підтримку всіх етапів життєвого циклу програмного забезпечення. Його використання пов’язується з можливістю організації узгодженого робочого процесу, у межах якого реалізується проєктування, розробка, тестування та супровід програмних продуктів без необхідності частого перемикання між різними інструментами [14]-[15].

Особливістю цього середовища є глибока інтеграція із сучасними засобами керування проєктами та контролю версій, що дозволяє ефективно організовувати колективну розробку. Забезпечується зручна взаємодія між учасниками команди, відстеження змін у коді та контроль якості виконуваних робіт. Це сприяє підвищенню узгодженості дій та зменшенню ризику виникнення помилок у складних програмних системах [14]-[15].

Важливим аспектом є підтримка широкого спектра мов програмування та технологій, що робить середовище універсальним інструментом для створення різнорідних програмних рішень. У межах одного проєкту може здійснюватися поєднання різних підходів і платформ, що відкриває

можливості для розробки складних систем із багаторівневою архітектурою. Така універсальність середовища особливо корисна у випадках, коли необхідно інтегрувати обробку даних, користувацькі інтерфейси та серверні компоненти [14]-[15].

Значну увагу приділено засобам аналізу якості коду, які дозволяють виявляти потенційні проблеми ще на етапі розробки. Автоматизовані перевірки, підказки щодо покращення структури коду та механізми рефакторингу сприяють формуванню більш ефективних і зрозумілих програмних рішень. Це має особливе значення для довготривалих проєктів, де підтримуваність коду є критичним фактором [14]-[15].

Необхідно також відзначити високий рівень адаптивності середовища до потреб користувача. Налаштування інтерфейсу, гарячих клавіш, тем оформлення та робочих просторів дозволяє оптимізувати процес розробки відповідно до індивідуальних вимог. Це підвищує продуктивність праці та зменшує навантаження під час виконання складних задач [14]-[15].

Окремо слід підкреслити підтримку сучасних підходів до розробки, зокрема використання контейнеризації, хмарних сервісів і засобів безперервної інтеграції. Це дозволяє розширити можливості середовища за межі локальної розробки та забезпечити повноцінну взаємодію з інфраструктурою розгортання програмних продуктів. У результаті формується єдиний технологічний простір, у якому реалізується повний цикл створення програмного забезпечення [14]-[15].

Таким чином, Visual Studio виступає як потужний інструмент, що поєднує у собі гнучкість, масштабованість і високий рівень автоматизації процесів розробки. Його застосування дозволяє підвищити ефективність створення програмних систем, забезпечити їх якість та спростити подальший супровід і розвиток [14]-[15].

Доцільність вибору середовища розробки Visual Studio для створення програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання зумовлюється його комплексним характером та

високим рівнем інтеграції інструментів, необхідних для реалізації подібних задач. У межах цього середовища забезпечується поєднання редактора коду, засобів компіляції, налагодження та тестування, що дозволяє організувати повний цикл розробки програм без необхідності залучення сторонніх інструментів [14]-[15].

Однією з визначальних переваг є розвинена підтримка мови C#, яка реалізується через інтелектуальні механізми аналізу коду. Автоматичне доповнення, перевірка синтаксису в режимі реального часу та рекомендації щодо оптимізації сприяють зменшенню кількості помилок і підвищенню якості програмного продукту. Це має особливе значення у випадку реалізації складної логіки перетворення математичних конструкцій, де точність і коректність є критичними [14]-[15].

Суттєву роль відіграють і потужні засоби налагодження, що дозволяють детально відстежувати виконання програми, аналізувати значення змінних та виявляти логічні помилки. У задачах, пов’язаних із парсингом тексту та побудовою формалізованих моделей, це дає змогу ефективно контролювати кожен етап обробки даних і своєчасно виявляти некоректні перетворення [14]-[15].

Важливою характеристикою є підтримка роботи з великими проєктами, що включає зручну організацію структури рішень, керування залежностями та інтеграцію з системами контролю версій. Це дозволяє забезпечити масштабованість програмної системи та спростити її подальший розвиток. Крім того, середовище підтримує автоматизоване тестування, що сприяє підвищенню надійності та стабільності програмного забезпечення [14]-[15].

Не менш значущими є можливості розширення функціональності через підключення додаткових компонентів і плагінів. Це дозволяє адаптувати середовище під специфічні потреби, зокрема для роботи з текстовими даними, математичними виразами або інструментами аналізу. Така гнучкість є важливою для реалізації спеціалізованих програмних систем [14]-[15].

Інтерфейс користувача середовища сприяє ефективній організації робочого процесу. Інструменти навігації, пошуку та рефакторингу коду дозволяють швидко орієнтуватися у великих обсягах програмного коду, що є важливим при роботі зі складними алгоритмами обробки інформації [14]-[15].

Таким чином, використання Visual Studio як середовища розробки є обґрунтованим завдяки поєднанню інтелектуальних засобів програмування, потужних інструментів налагодження, підтримки масштабованих проєктів та можливостей розширення. Це створює сприятливі умови для ефективної реалізації програмної системи перетворення математичних конструкцій у кодове подання та забезпечує високий рівень якості кінцевого продукту.

Обґрунтування вибору середовища розробки для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	SharpDevelop	MonoDevelop
Повнота функціоналу	+	–	+–
Продуктивність середовища	+	+–	+–
Підтримка тестування	+	–	+–
Зручність інтерфейсу	+	+	+–
Підтримка розширень	+	–	+–
Інтеграція з платформою .NET	+	+–	+–

Таким чином, для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано середовище розробки Visual Studio, що надає інтелектуальні засоби програмування, потужні інструменти налагодження, підтримку масштабованих проєктів. Не менш значущими є можливості розширення функціональності через підключення додаткових компонентів і плагінів, що дозволяє адаптувати середовище під специфічні потреби, зокрема для роботи з текстовими даними, математичними виразами або інструментами аналізу. Це створює сприятливі умови для ефективної реалізації програмної системи перетворення математичних конструкцій у кодове подання та забезпечує високий рівень якості кінцевого продукту.

2.3 Вибір засобів зберігання та опрацювання даних

Для зберігання та опрацювання даних про математичні конструкції з текстових файлів з подальшим їх перетворенням у кодове подання було обрано мову XML [16]-[17].

Доцільність використання XML-подібної мови для представлення математичних рівнянь і формул у межах програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання пояснюється її здатністю забезпечувати чітку структуризацію складних даних. Математичні вирази за своєю природою мають ієрархічну будову, у якій елементи підпорядковуються один одному відповідно до правил пріоритету операцій і вкладеності. XML-подібний підхід дозволяє природно відобразити таку структуру у вигляді деревоподібних конструкцій, що значно спрощує подальший аналіз і трансформацію [16]-[17].

Важливим аспектом є формалізованість такого подання, яка забезпечує однозначність інтерпретації. На відміну від звичайного текстового запису, де можливі різночитання або неоднозначності, XML-подібна мова задає строгі правила опису кожного елемента. Це дозволяє уникнути помилок, пов’язаних

із некоректним розпізнаванням математичних конструкцій, і створює надійну основу для автоматизованого перетворення у програмний код [16]-[17].

Суттєвою перевагою є також сумісність із численними інструментами обробки даних. XML-подібні формати підтримуються широким спектром бібліотек і технологій, що дозволяє ефективно реалізовувати парсинг, валідацію та трансформацію даних. Завдяки цьому забезпечується можливість використання готових рішень для аналізу структури документів, що скорочує час розробки та підвищує надійність системи [16]-[17].

Окремо слід підкреслити зручність розширення такого формату. У процесі розвитку програмної системи може виникати необхідність підтримки нових типів математичних конструкцій або спеціалізованих операцій. XML-подібна мова дозволяє без суттєвих змін у базовій архітектурі додавати нові елементи та атрибути, зберігаючи при цьому сумісність із уже існуючими даними. Це створює передумови для довготривалого використання системи та її адаптації до нових вимог [16]-[17].

Не менш важливою є можливість відокремлення представлення даних від логіки їх обробки. XML-подібний формат виступає як універсальний проміжний шар, що дозволяє розділити етапи розпізнавання, аналізу та генерації коду. Такий підхід підвищує гнучкість системи та спрощує її модифікацію, оскільки зміни в одному компоненті не потребують суттєвого втручання в інші частини [16]-[17].

Крім того, використання подібного формату сприяє підвищенню прозорості та зрозумілості представлення математичних виразів. Це важливо як для розробників, так і для користувачів системи, оскільки дозволяє легко перевіряти правильність інтерпретації та при необхідності вносити корективи. У поєднанні з можливістю автоматичної валідації це забезпечує додатковий рівень контролю якості обробки даних [16]-[17].

Обґрунтування вибору засобів зберігання та опрацювання даних наведено у таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору засобів зберігання та опрацювання даних

Критерій порівняння засобів зберігання даних	Засоби зберігання даних		
	XML	JSON	LaTeX
Ієрархічне представлення математичних конструкцій	+	+–	+–
Формалізованість і строгість структури	+	+–	–
Зручність автоматичного парсингу	+	+	–
Підтримка валідації структури	+	+–	+–
Зручність представлення складних формул	+	+–	+

Таким чином, для зберігання та опрацювання даних у вигляді математичних рівнянь і формул обрано мову XML завдяки її здатності забезпечувати структурованість, однозначність, розширюваність і сумісність із сучасними технологіями обробки даних. Це робить її ефективним інструментом для побудови надійних і гнучких програмних систем перетворення математичних конструкцій у кодове подання.

2.4 Висновки за розділом 2

Для реалізації програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано мову програмування C#, що забезпечує широкий набір вбудованих засобів для роботи з текстом, файлами та структурами даних. Це дозволяє ефективно реалізовувати складні алгоритми аналізу та обробки текстових джерел, у яких містяться математичні

конструкції. Наявність потужних бібліотек для роботи з регулярними виразами, потоками даних і колекціями значно спрощує розробку компонентів, відповідальних за синтаксичний аналіз і трансформацію інформації.

Для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано середовище розробки Visual Studio, що надає інтелектуальні засоби програмування, потужні інструменти налагодження, підтримку масштабованих проєктів. Не менш значущими є можливості розширення функціональності через підключення додаткових компонентів і плагінів, що дозволяє адаптувати середовище під специфічні потреби, зокрема для роботи з текстовими даними, математичними виразами або інструментами аналізу. Це створює сприятливі умови для ефективної реалізації програмної системи перетворення математичних конструкцій у кодове подання та забезпечує високий рівень якості кінцевого продукту.

Для зберігання та опрацювання даних у вигляді математичних рівнянь і формул обрано мову XML завдяки її здатності забезпечувати структурованість, однозначність, розширюваність і сумісність із сучасними технологіями обробки даних. Це робить її ефективним інструментом для побудови надійних і гнучких програмних систем перетворення математичних конструкцій у кодове подання.

3 ОПИС ПРОГРАМНОЇ СИСТЕМИ

3.1 Структура програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Структуру програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання наведено на рис. 3.1.

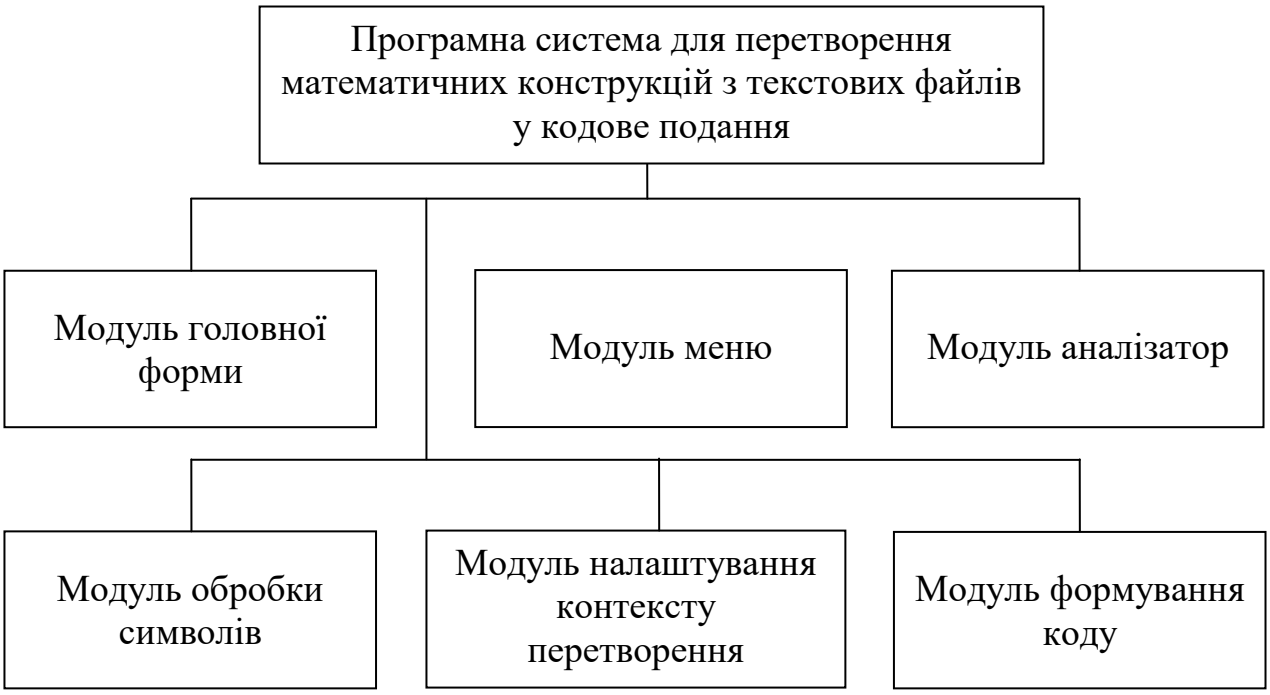


Рисунок 3.1 – Структура програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Структура програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання являє собою сукупність взаємопов’язаних модулів, кожен із яких відповідає за окремий етап обробки даних. Такий підхід дозволяє забезпечити чіткий розподіл відповідальності, підвищити керованість програмної системи та спростити її подальший розвиток. Розроблено шість основних модулів (модуль головної форми, модуль меню, модуль аналізатор, модуль обробки символів, модуль налаштування контексту перетворення, модуль формування коду), що

реалізують повний цикл трансформації математичних виразів від текстового представлення до програмного коду.

Модуль головної форми призначено для організації взаємодії користувача з програмною системою у вигляді окремого програмного застосування. У цьому модулі забезпечується введення текстових даних, ініціювання процесу перетворення та відображення отриманих результатів. Його функціональність охоплює керування основними сценаріями роботи, обробку подій користувача та забезпечення доступу до внутрішніх механізмів конвертації. Через цей модуль реалізується централізований контроль над виконанням операцій.

Модуль меню виконує аналогічні функції у контексті інтеграції з текстовим редактором, де програмна система представлена у вигляді надбудови. У цьому випадку взаємодія з користувачем здійснюється через елементи інтерфейсу стрічки, що дозволяє запускати процедури перетворення безпосередньо в середовищі роботи з документами. Забезпечується доступ до функцій конвертації, обробка команд та передача даних до інших компонентів системи.

Модуль аналізатор орієнтується на розбір текстового представлення математичних конструкцій. У межах цього модуля здійснюється синтаксичний аналіз вхідних даних, виділення структурних елементів і побудова внутрішнього формалізованого подання, що відповідає XML-подібній моделі. Реалізується інтерпретація символів, операторів і функцій, а також формування ієрархічної структури, яка відображає логіку математичного виразу.

Модуль обробки символів виконує допоміжну функцію, пов’язану з нормалізацією та уніфікацією вхідних даних. Особлива увага приділяється коректній обробці специфічних символів, зокрема літер грецького алфавіту, які часто використовуються у математичних формулах. Забезпечується перетворення таких символів у стандартизоване представлення, що дозволяє уникнути неоднозначностей на подальших етапах аналізу.

Модуль налаштування контексту перетворення відповідає за опрацювання параметрів, що визначають правила трансформації математичних конструкцій у кодове подання. У межах цього модуля здійснюється формування конфігурацій, які враховують особливості цільової мови програмування, типи даних, формат вихідного коду та інші аспекти, що впливають на результат перетворення. Це дозволяє адаптувати систему до різних умов використання без зміни її базової логіки.

Модуль формування коду є завершальним етапом обробки, у якому відбувається безпосереднє перетворення внутрішнього представлення математичних конструкцій у програмний код. Використовуючи дані, сформовані попередніми модулями, цей компонент здійснює генерацію синтаксично коректних конструкцій мовою програмування, зокрема C#. Реалізується обхід ієрархічної структури, побудова виразів та формування кінцевого результату, придатного для подальшого використання у програмному середовищі.

Узгоджена взаємодія зазначених модулів забезпечує функціонування системи як цілісного механізму, у якому кожен компонент виконує чітко визначену роль. Така архітектура дозволяє ефективно обробляти математичні конструкції, забезпечуючи їх коректне перетворення та адаптацію до програмного подання. Отже, описана структура програмної системи відзначається логічною завершеністю та модульною організацією, що сприяє підвищенню надійності, гнучкості та масштабованості рішення, а також створює сприятливі умови для його подальшого розвитку та вдосконалення.

3.2 Функціонування програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Функціонування програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання подамо за допомогою схеми, зображеної на рис. 3.2.

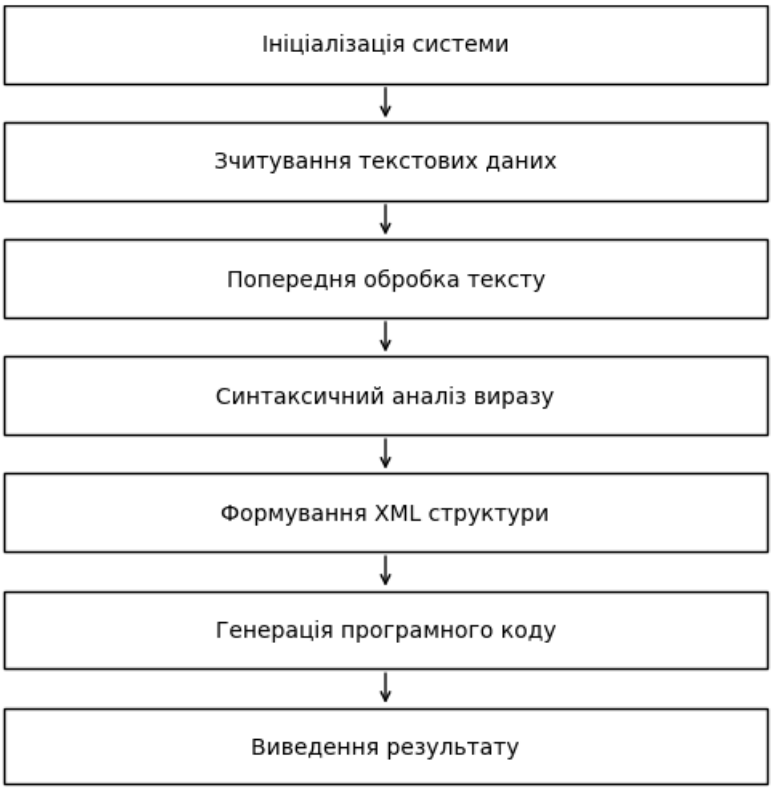


Рисунок 3.2 – Схема функціонування програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Послідовність функціонування програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання представлено як впорядкований процес обробки даних, у якому кожен етап логічно переходить у наступний.

На початковому етапі здійснюється запуск програмної системи та ініціалізація її основних компонентів, після чого відбувається вибір джерела вхідних даних. У ролі такого джерела виступає текстовий файл або фрагмент документа, що містить математичні рівняння чи формули. Після отримання доступу до вхідних даних виконується їх зчитування та попередня підготовка, яка передбачає усунення зайвих символів, нормалізацію форматування та приведення запису до стандартизованого вигляду.

Далі запускається етап синтаксичного аналізу, у межах якого текстове представлення математичних конструкцій розбирається на окремі складові елементи. Визначаються оператори, операнди, функції та спеціальні символи,

а також встановлюються зв’язки між ними. На основі цього формується внутрішнє структуроване подання, яке відповідає XML-подібній моделі та відображає ієрархію математичного виразу.

Після побудови структури виконується етап семантичної обробки, під час якого уточнюється зміст елементів, перевіряється коректність їх поєднання та усуваються можливі неоднозначності. Одночасно здійснюється обробка специфічних символів, зокрема перетворення літер грецького алфавіту та інших нестандартних позначень у внутрішньо узгоджене представлення.

Наступним кроком є формування контексту перетворення, що включає визначення параметрів генерації коду. На цьому етапі враховуються особливості цільової мови програмування, типи змінних, правила запису математичних операцій і загальний формат результату. Підготовлений контекст використовується для узгодження структури даних із вимогами до кінцевого програмного подання.

Після цього виконується безпосереднє перетворення структурованої моделі у програмний код. Реалізується обхід ієрархічного представлення математичної конструкції, під час якого кожен елемент трансформується у відповідний фрагмент коду. У результаті формується завершений вираз або набір інструкцій, що відповідають синтаксису обраної мови програмування.

Завершальний етап передбачає виведення отриманого результату користувачеві або його збереження у файл. За необхідності може виконуватися додаткова перевірка коректності згенерованого коду, після чого процес вважається завершеним.

Отже, функціонування програмної системи являє собою послідовний і логічно узгоджений процес переходу від неструктурованого текстового представлення математичних конструкцій до формалізованого програмного коду, що забезпечує точність, відтворюваність і ефективність обробки даних.

3.3 Опис структури даних для подання математичних конструкцій

Структура даних для подання математичних конструкцій у програмній системі, що здійснює перетворення з текстових файлів у кодове подання, являє собою формалізовану ієрархічну модель, побудовану на основі XML-подібного підходу. У такій моделі кожен математичний вираз інтерпретується як дерево, у якому вузли відповідають операціям, функціям або операндам, а зв’язки між ними відображають порядок виконання дій і вкладеність. Завдяки цьому забезпечується точне відтворення логіки математичного запису незалежно від його початкового текстового представлення.

Основою структури виступає кореневий елемент, який визначає межі окремої математичної конструкції та містить усі її складові. Усередині нього розміщуються вкладені елементи, що описують операції, змінні, константи та функції. Кожен елемент має чітко визначене семантичне призначення, що дозволяє однозначно трактувати його під час подальшої обробки. Атрибути елементів можуть містити додаткову інформацію, наприклад тип даних, ідентифікатори або параметри функцій, що розширює можливості представлення складних математичних структур.

Особливістю такої організації є те, що порядок виконання операцій задається не лише позиційно, а й структурно, через вкладеність елементів. Це дозволяє уникнути неоднозначностей, характерних для лінійного текстового запису, і значно спрощує побудову алгоритмів аналізу. У процесі перетворення в кодове подання така структура може безпосередньо трансформуватися у відповідні конструкції мови програмування, оскільки кожен вузол дерева відповідає конкретній операції або значенню.

Крім того, у межах XML-структури забезпечується можливість розширення набору підтримуваних математичних елементів. Нові типи операцій або функцій можуть бути додані без зміни базової логіки обробки, що досягається за рахунок використання узагальнених правил інтерпретації.

Це створює гнучке середовище для розвитку програмної системи та адаптації до нових задач.

Приклад подання математичного виразу за допомогою структури даних на основі мови XML наведено на рис. 3.3.

```
<Expression>
    <Operation type="add">
        <Operand>
            <Variable name="x"/>
        </Operand>
        <Operand>
            <Operation type="multiply">
                <Operand>
                    <Constant value="2"/>
                </Operand>
                <Operand>
                    <Variable name="y"/>
                </Operand>
            </Operation>
        </Operand>
    </Operation>
</Expression>
```

Рисунок 3.3 – Приклад подання математичного виразу за допомогою структури даних на основі мови XML

У прикладі на рис. 3.3 відображено вираз вигляду x + 2 * y. Кореневий елемент визначає всю конструкцію, всередині якої знаходиться операція додавання. Перший операнд цієї операції є змінною x, тоді як другий операнд представлено як вкладену операцію множення. У межах цієї вкладеної операції один елемент відповідає константі зі значенням 2, а інший – змінній y. Таким чином, структура чітко відображає пріоритет виконання дій,

де множення виконується раніше за додавання, що задається вкладеністю елементів.

Завдяки такому підходу забезпечується можливість прямого обходу структури та генерації відповідного програмного коду, наприклад у вигляді виразів конкретної мови програмування. Кожен елемент може бути оброблений як окремий вузол абстрактного синтаксичного дерева, що спрощує реалізацію алгоритмів трансформації.

Отже, описана XML-орієнтована структура даних дозволяє ефективно представляти математичні конструкції у формалізованому вигляді, забезпечуючи їх однозначність, розширюваність та зручність для автоматизованої обробки. Це створює надійну основу для побудови програмних систем, здатних коректно та гнучко перетворювати текстові математичні вирази у кодове подання.

3.4 Проєктування інтерфейсу програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Проєктування інтерфейсу взаємодії користувача з програмною системою для перетворення математичних конструкцій з текстових файлів у кодове подання виконано з урахуванням вимог технічного завдання. При розробленні програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання враховані функціональні вимоги до програми:

– підтримка можливості зчитування математичних конструкцій із текстових файлів та їх попереднього опрацювання для подальшого аналізу;

– підтримка можливості синтаксичного розбору математичних виразів із урахуванням операторів, функцій, змінних та вкладених структур;

– підтримка можливості формування внутрішнього представлення математичних конструкцій на основі XML-подібної моделі;

– підтримка можливості обробки спеціальних символів, зокрема літер грецького алфавіту та інших математичних позначень;

– можливість генерації програмного коду на основі структурованого представлення математичних виразів;

– підтримка можливості коректного врахування пріоритетів операцій та вкладеності виразів під час генерації коду;

– можливість відображення результатів перетворення у зручному для користувача вигляді в інтерфейсі програми.

Інтерфейс головної форми програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання наведено на рис. 3.4.

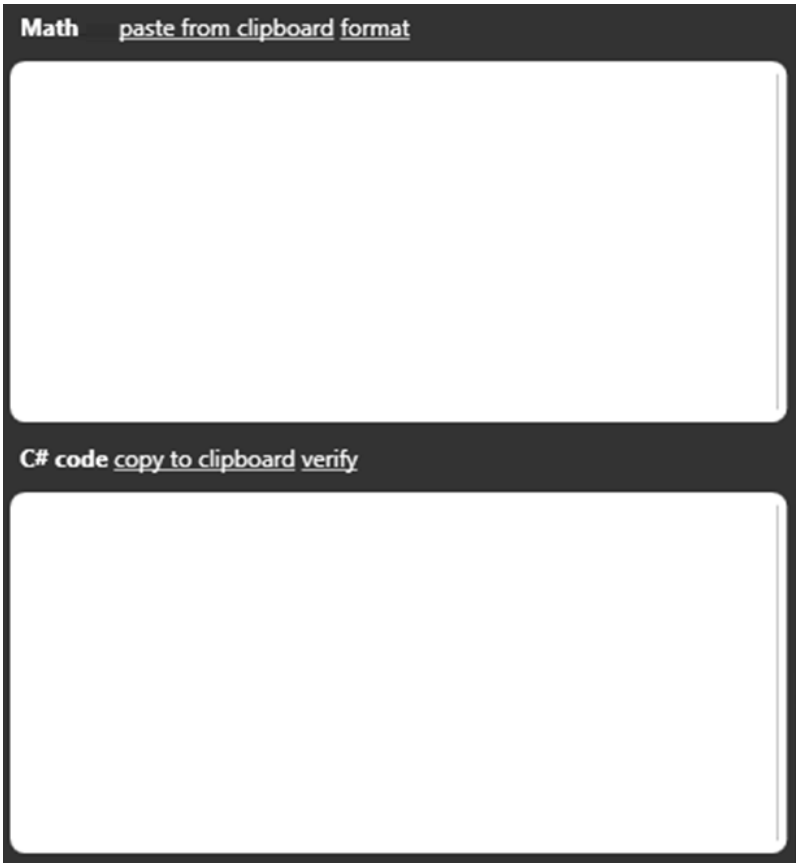


Рисунок 3.4 – Інтерфейс головної форми програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Організація інтерфейсу ґрунтується на принципі розділення робочого простору на функціональні області, кожна з яких відповідає певному аспекту взаємодії. Передбачається наявність зони введення, де розміщується текстове представлення математичних конструкцій, а також області відображення результатів, у якій формується програмний код. Такий підхід дозволяє візуально розмежувати вхідні та вихідні дані, що спрощує сприйняття процесу перетворення та дає можливість швидко оцінити його результат.

Окремої уваги потребує відображення результатів обробки та інформування користувача про стан виконання операцій. Інтерфейс програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання забезпечує зрозуміле представлення згенерованого коду, а також повідомлення про успішність або помилки під час виконання перетворення. Це сприяє підвищенню довіри до системи та дозволяє оперативно реагувати на можливі неточності.

Таким чином, інтерфейс програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання сформовано як цілісне середовище взаємодії, у якому поєднуються зручність, функціональність і зрозумілість.

3.5 Висновки за розділом 3

Розроблено програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання.

Запропоновано структуру програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Визначено, що структура програмної системи являє собою сукупність взаємопов’язаних модулів, кожен із яких відповідає за окремий етап обробки даних. Розроблено шість основних модулів (модуль головної форми, модуль меню, модуль аналізатор, модуль обробки символів, модуль налаштування контексту перетворення, модуль формування коду), що реалізують повний цикл

трансформації математичних виразів від текстового представлення до програмного коду. Такий підхід дозволяє забезпечити чіткий розподіл відповідальності, підвищити керованість програмної системи та спростити її подальший розвиток.

Послідовність функціонування програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання представлено як впорядкований процес обробки даних, у якому кожен етап логічно переходить у наступний. Визначено, що функціонування програми являє собою послідовний і логічно узгоджений процес переходу від неструктурованого текстового представлення математичних конструкцій до формалізованого програмного коду, що забезпечує точність, відтворюваність і ефективність обробки даних.

Для подання математичних конструкцій у розробленій програмній системі використано структуру даних, що являє собою формалізовану ієрархічну модель, побудовану на основі XML-подібного підходу. У такій моделі кожен математичний вираз інтерпретується як дерево, у якому вузли відповідають операціям, функціям або операндам, а зв’язки між ними відображають порядок виконання дій і вкладеність. Завдяки цьому забезпечується точне відтворення логіки математичного запису незалежно від його початкового текстового представлення. Це дозволяє ефективно представляти математичні конструкції у формалізованому вигляді, забезпечуючи їх однозначність, розширюваність та зручність для автоматизованої обробки.

Виконано проєктування інтерфейсу взаємодії користувача з програмною системою для перетворення математичних конструкцій з текстових файлів у кодове подання.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Призначення й умови застосування програмної системи

Програмна система призначена для підтримки процесу перетворення математичних конструкцій з текстових файлів у кодове подання. Програмна система написана на мові C#, що забезпечує широкий набір вбудованих засобів для роботи з текстом, файлами та структурами даних. В якості середовища розробки для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано Visual Studio, що надає інтелектуальні засоби програмування, потужні інструменти налагодження, підтримку масштабованих проєктів.

Програмна система для перетворення математичних конструкцій з текстових файлів у кодове подання забезпечує виконання таких функцій:

– підтримка можливості зчитування математичних конструкцій із текстових файлів та їх попереднього опрацювання для подальшого аналізу;

– підтримка можливості синтаксичного розбору математичних виразів із урахуванням операторів, функцій, змінних та вкладених структур;

– підтримка можливості формування внутрішнього представлення математичних конструкцій на основі XML-подібної моделі;

– підтримка можливості обробки спеціальних символів, зокрема літер грецького алфавіту та інших математичних позначень;

– можливість генерації програмного коду на основі структурованого представлення математичних виразів;

– підтримка можливості коректного врахування пріоритетів операцій та вкладеності виразів під час генерації коду;

– можливість відображення результатів перетворення у зручному для користувача вигляді в інтерфейсі програми.

Для роботи програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання потрібні такі технічні та

програмні ресурси: персональний комп’ютер або ноутбук з процесором, тактова частота якого не нижча за 2.0 ГГц, що дозволяє ефективно виконувати операції синтаксичного аналізу та генерації коду, обсяг оперативної пам’яті має становити не менше 4 ГБ. Для зберігання програмного забезпечення та оброблюваних даних необхідно не менше 500 МБ вільного місця на накопичувачі, при цьому використання твердотільного диска сприяє підвищенню швидкодії системи. Також передбачається наявність стандартних пристроїв введення та виведення інформації, таких як клавіатура, миша та дисплей.

З боку програмного забезпечення необхідною є наявність операційної системи сімейства Windows, починаючи з версії Windows 10 або новішої, що забезпечує підтримку сучасних засобів розробки та виконання програм. Обов’язковою умовою є встановлення платформи .NET відповідної версії, яка використовується для виконання програм, розроблених мовою C#. Для модифікації програмного продукту може знадобитися середовище розробки або виконання, таке як Visual Studio, що забезпечує підтримку необхідних бібліотек і компонентів. У випадку використання інтеграції з текстовим редактором додатково передбачається встановлення Microsoft Word, у якому реалізовано функціональність надбудови для роботи з математичними конструкціями.

Окрім цього, важливою є наявність стандартних бібліотек для обробки XML-подібних структур, що забезпечують коректний парсинг і трансформацію даних.

Функціональні характеристики розробленої програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання наведено у технічному завданні (додаток А). Фрагмент тексту програмного коду програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання наведено у додатку Б.

4.2 Характеристики програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Програмна система перетворення математичних конструкцій з текстових файлів у кодове подання характеризується як спеціалізований інструмент, орієнтований на автоматизацію обробки формалізованих знань та забезпечення переходу від текстового опису до програмної реалізації. Її функціонування ґрунтується на поєднанні методів синтаксичного аналізу, структурованого представлення даних і генерації коду, що дозволяє забезпечити цілісність і логічну узгодженість результатів.

Суттєвою характеристикою програмної системи є здатність інтерпретувати математичні вирази незалежно від їх початкової форми подання. Завдяки цьому забезпечується гнучкість у роботі з різними джерелами текстових даних, що можуть відрізнятися за структурою, стилем запису та використаними позначеннями. У процесі обробки досягається уніфікація вхідної інформації, що дозволяє звести різноманіття записів до єдиного формалізованого вигляду.

Важливою властивістю є використання проміжного структурованого представлення, яке відображає внутрішню логіку математичних конструкцій. Такий підхід дозволяє відокремити етап аналізу від етапу генерації коду, що сприяє підвищенню гнучкості системи та спрощує її адаптацію до різних мов програмування. У результаті формується універсальний механізм трансформації, який може бути розширений без суттєвих змін у базовій архітектурі.

Програмна система також характеризується високим рівнем точності відтворення математичних залежностей. Забезпечується коректне врахування пріоритетів операцій, вкладеності виразів і взаємозв’язків між їх складовими. Це дозволяє мінімізувати ризик логічних помилок під час перетворення та гарантувати відповідність отриманого коду вихідному математичному змісту.

Програма може бути інтегрована з іншими програмними середовищами, що забезпечує універсальність застосування та розширює сферу використання, зокрема у навчальних, наукових і прикладних задачах.

Характерною рисою є також підтримка модульної архітектури, що сприяє структурованості внутрішньої організації та спрощує процес супроводу. Кожен компонент виконує визначену функцію, а взаємодія між ними здійснюється через чітко визначені інтерфейси. Такий підхід забезпечує можливість поступового вдосконалення системи без порушення її загальної логіки.

Таким чином, програмна система вирізняється поєднанням гнучкості, формалізованості та здатності до інтеграції, що робить її ефективним засобом автоматизованого перетворення математичних конструкцій у програмне подання та сприяє підвищенню продуктивності роботи з формалізованими даними.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання необхідно запустити відповідний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмної системи є математичні конструкції (формули, рівняння) у текстових файлах.

Вихідними даними програмної системи є кодове подання відповідних математичних конструкцій.

4.3.3 Повідомлення

Користувач програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання може отримати такі повідомлення: повідомлення про некоректність введених вхідних даних.

4.4 Виконання програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Після запуску програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання користувачу відображається головна форма (рис. 4.1).

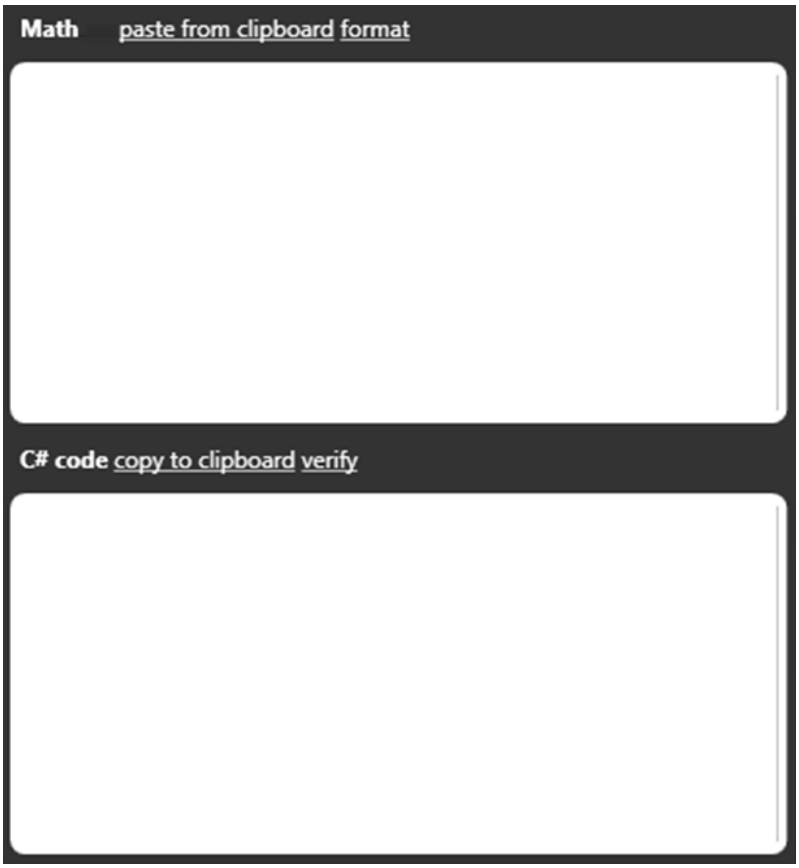


Рисунок 4.1 – Інтерфейс головної форми програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Після ініціалізації програмного застосування передбачається використання буфера обміну як джерела вхідних даних. Для цього виконується вставлення попередньо скопійованого тексту математичного виразу (наприклад, $y = ax^2 + bx + c$) з текстового документу шляхом активації відповідної функції, що відповідає за імпорт даних.

У результаті виконання цієї дії у верхній області інтерфейсу відображається текстове представлення рівняння, тоді як у нижній частині автоматично формується його інтерпретація у вигляді програмного коду мовою C# (рис. 4.2).

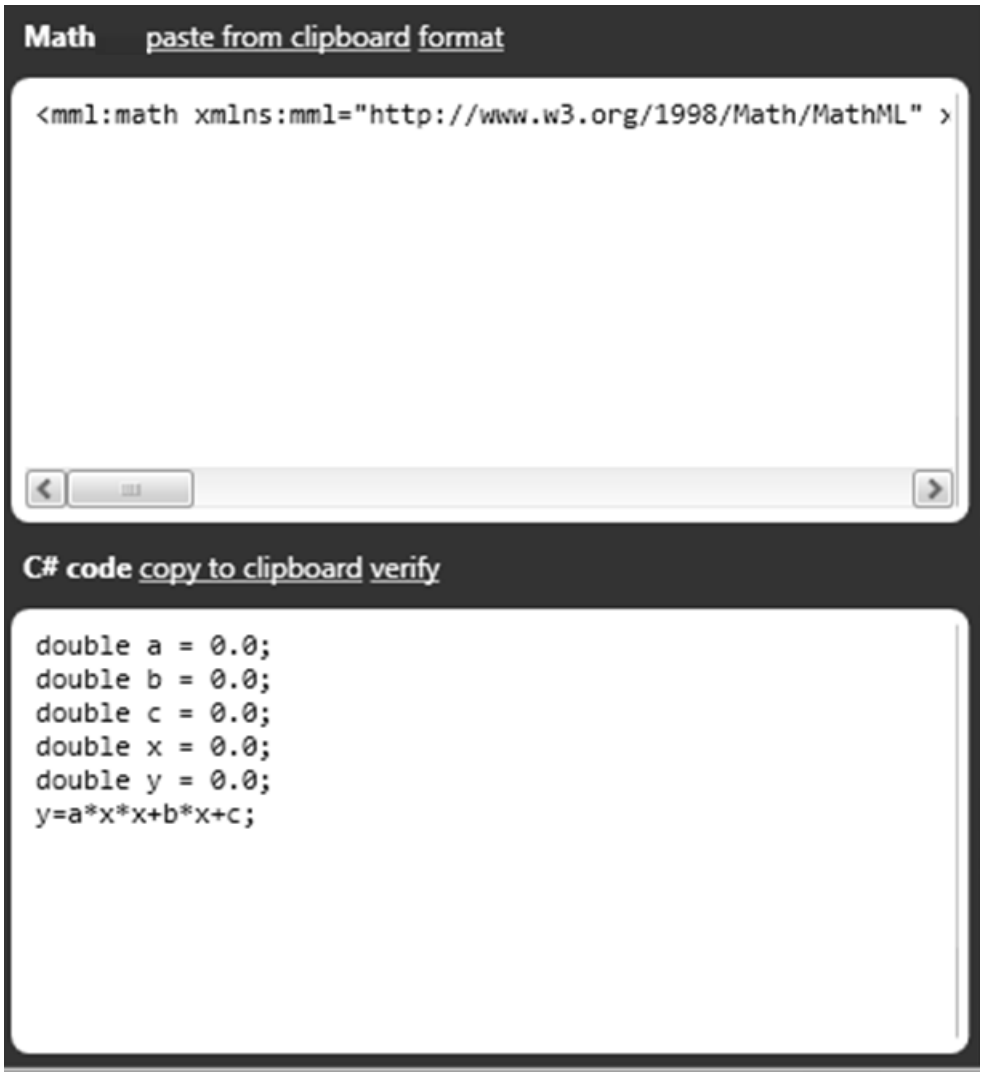


Рисунок 4.2 – Перетворення математичної формули з текстового подання у програмний код

Після завершення процесу перетворення забезпечується можливість подальшої роботи з отриманим результатом. Зокрема, передбачено перенесення згенерованого коду до буфера обміну для його використання в інших програмних середовищах. Окрім цього, реалізовано механізм перевірки коректності сформованого коду, що дозволяє оцінити правильність виконаного перетворення та виявити можливі неточності. У разі, якщо перевірка завершується без виявлення помилок, система інформує користувача про успішне проходження цього етапу шляхом виведення відповідного повідомлення (рис. 4.3). Такий підхід забезпечує додатковий контроль якості та підвищує надійність отриманих результатів.

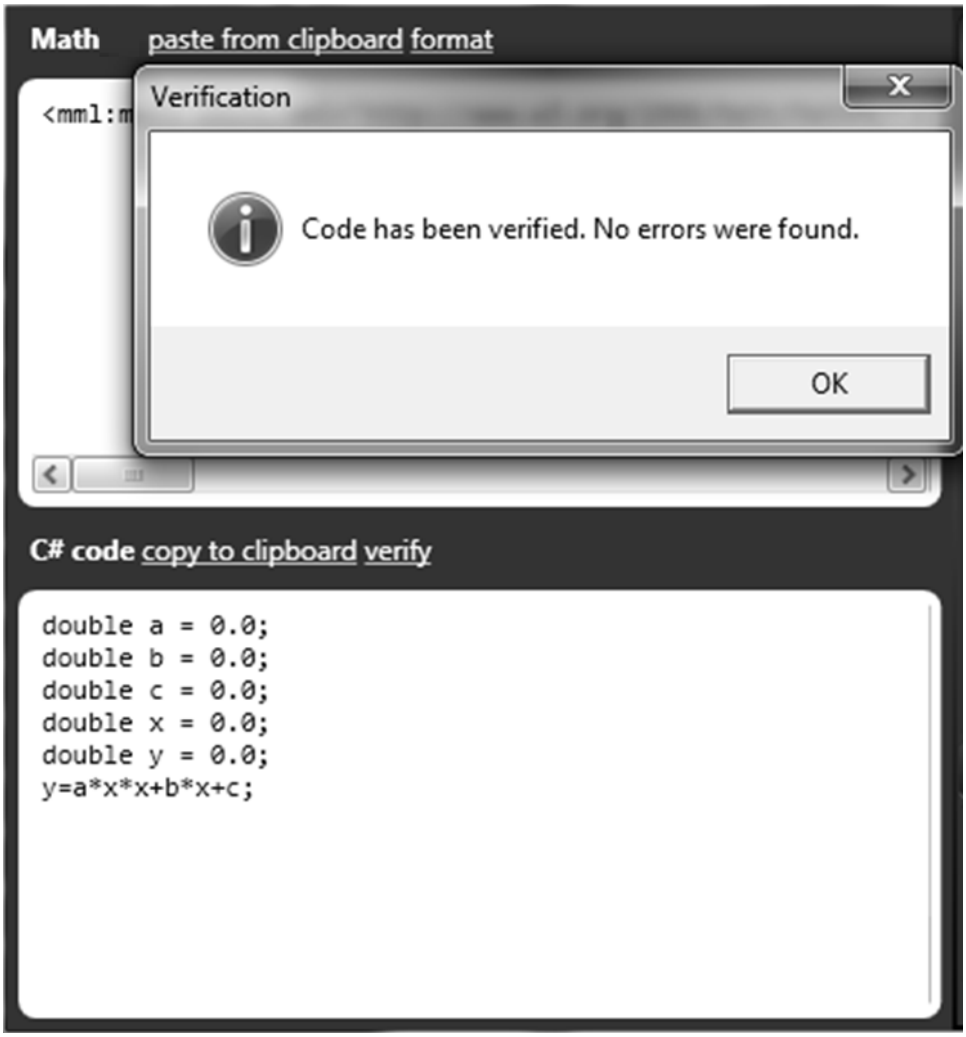


Рисунок 4.3 – Перевірка програмного коду,
що відповідає математичному виразу

За потреби інтеграції функціональності програми у середовище текстового редактора передбачено встановлення спеціалізованої надбудови. Це досягається шляхом запуску інсталяційного пакета, після чого відповідний модуль додається до стрічки інструментів застосування Microsoft Word (рис. 4.4). У результаті функції перетворення математичних конструкцій стають доступними безпосередньо під час роботи з документами, що значно спрощує їх використання у практичній діяльності.

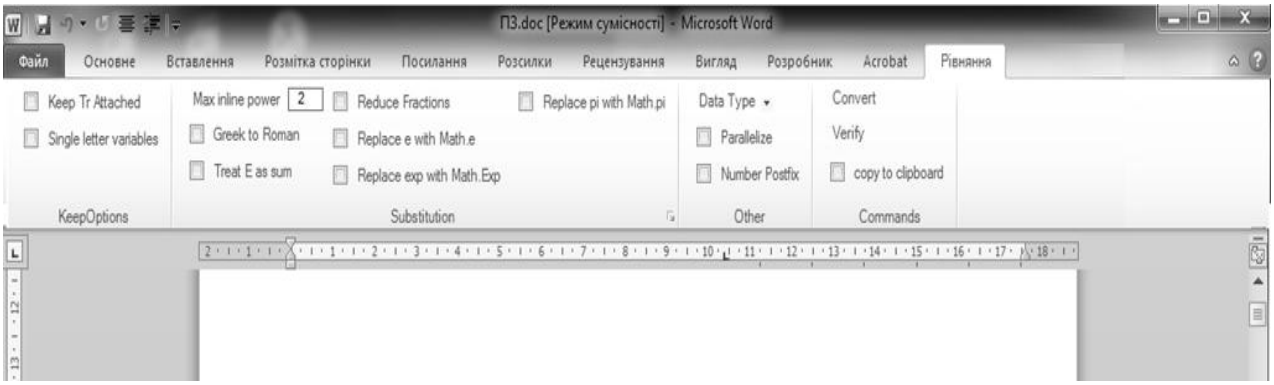


Рисунок 4.4 – Додавання надбудови програмної системи у Microsoft Word

4.5 Тестування програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання

Проведено тестування програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання.

Підтверджено, що програмна система для перетворення математичних конструкцій з текстових файлів у кодове подання працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги та успішно справляючись із основною задачею підтримки процесу взаємодії користувачів.

Розроблена система забезпечує організацію та опрацювання даних, пов’язаних із підтримкою процесів конвертації математичних конструкцій у кодове подання.

4.6 Висновки за розділом 4

Описано програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання.

Проведено тестування створеної програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов’язаних із підтримкою процесів конвертації математичних конструкцій у кодове подання.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процеси, пов’язані зі створенням програмних систем для перетворення даних.

Визначено, що програмна система перетворення математичних конструкцій з текстових файлів у кодове подання являє собою спеціалізований програмний інструмент, призначений для автоматизованого переходу від текстового опису математичних виразів до їх формалізованого представлення у вигляді програмного коду. У межах такого підходу забезпечується інтерпретація математичних записів, їх структуризація та подальше відтворення у вигляді синтаксично коректних конструкцій певної мови програмування. Основною метою функціонування подібних систем виступає спрощення роботи з математичними даними та підвищення ефективності їх використання у програмному середовищі.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних систем для перетворення даних. Проте деякі програмні системи перетворення даних орієнтовані на конкретні формати вхідних даних або певні математичні нотації. У разі відхилення від передбачених шаблонів ефективність обробки істотно зменшується, а інколи процес перетворення стає неможливим без попередньої ручної підготовки тексту. Це частково нівелює переваги автоматизації та вимагає додаткових ресурсів. Проблеми виникають також на етапі семантичного аналізу, оскільки математичні конструкції часто мають неоднозначну інтерпретацію без урахування контексту. Крім того, розробка та підтримка таких програмних рішень вимагає залучення фахівців високого рівня, здатних поєднувати знання у галузях математики, програмування та обробки природної мови. Це ускладнює масштабування систем і підвищує вартість їх впровадження та супроводу. Тому актуальною є розробка програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання.

Сформульовано функціональні вимоги до програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання.

Для реалізації програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано мову програмування C#, що забезпечує широкий набір вбудованих засобів для роботи з текстом, файлами та структурами даних. Це дозволяє ефективно реалізовувати складні алгоритми аналізу та обробки текстових джерел, у яких містяться математичні конструкції. Наявність потужних бібліотек для роботи з регулярними виразами, потоками даних і колекціями значно спрощує розробку компонентів, відповідальних за синтаксичний аналіз і трансформацію інформації.

Для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано середовище розробки Visual Studio, що надає інтелектуальні засоби програмування, потужні інструменти налагодження, підтримку масштабованих проєктів. Не менш значущими є можливості розширення функціональності через підключення додаткових компонентів і плагінів, що дозволяє адаптувати середовище під специфічні потреби, зокрема для роботи з текстовими даними, математичними виразами або інструментами аналізу. Це створює сприятливі умови для ефективної реалізації програмної системи перетворення математичних конструкцій у кодове подання та забезпечує високий рівень якості кінцевого продукту.

Для зберігання та опрацювання даних у вигляді математичних рівнянь і формул обрано мову XML завдяки її здатності забезпечувати структурованість, однозначність, розширюваність і сумісність із сучасними технологіями обробки даних. Це робить її ефективним інструментом для побудови надійних і гнучких програмних систем перетворення математичних конструкцій у кодове подання.

Розроблено програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання.

Запропоновано структуру програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Визначено, що структура програмної системи являє собою сукупність взаємопов’язаних модулів, кожен із яких відповідає за окремий етап обробки даних. Розроблено шість основних модулів (модуль головної форми, модуль меню, модуль аналізатор, модуль обробки символів, модуль налаштування контексту перетворення, модуль формування коду), що реалізують повний цикл трансформації математичних виразів від текстового представлення до програмного коду. Такий підхід дозволяє забезпечити чіткий розподіл відповідальності, підвищити керованість програмної системи та спростити її подальший розвиток.

Послідовність функціонування програмної системи перетворення математичних конструкцій із текстових файлів у кодове подання представлено як впорядкований процес обробки даних, у якому кожен етап логічно переходить у наступний. Визначено, що функціонування програми являє собою послідовний і логічно узгоджений процес переходу від неструктурованого текстового представлення математичних конструкцій до формалізованого програмного коду, що забезпечує точність, відтворюваність і ефективність обробки даних.

Для подання математичних конструкцій у розробленій програмній системі використано структуру даних, що являє собою формалізовану ієрархічну модель, побудовану на основі XML-подібного підходу. У такій моделі кожен математичний вираз інтерпретується як дерево, у якому вузли відповідають операціям, функціям або операндам, а зв’язки між ними відображають порядок виконання дій і вкладеність. Завдяки цьому забезпечується точне відтворення логіки математичного запису незалежно від його початкового текстового представлення. Це дозволяє ефективно представляти математичні конструкції у формалізованому вигляді, забезпечуючи їх однозначність, розширюваність та зручність для автоматизованої обробки.

Виконано проєктування інтерфейсу взаємодії користувача з програмною системою для перетворення математичних конструкцій з текстових файлів у кодове подання.

Проведено тестування створеної програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов’язаних із підтримкою процесів конвертації математичних конструкцій у кодове подання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Parsing Expressions. URL: https://craftinginterpreters.com/parsing-expressions.html (date of access: 07.05.2026).

2. Writing a Mathematical Expression Parser. URL: https://itnext.io/writing-a-mathematical-expression-parser-35b0b78f869e (date of access: 07.05.2026).

3. A Working Mathematician's Guide to Parsing. URL: https://www.jeremykun.com/2019/04/20/a-working-mathematicians-guide-to-parsing (date of access: 07.05.2026).

4. Let’s build a simple math expression parser. URL: https://www.esveo.com/en/blog/let-s-build-a-simple-math-expression-parser/ (date of access: 07.05.2026).

5. Parsing mathematical expressions. URL: https://tristanpenman.com/blog/posts/2019/03/31/parsing-mathematical-expressions/ (date of access: 07.05.2026).

6. Mathpix. URL: https://mathpix.com/equation-to-word (date of access: 07.05.2026).

7. Bibcit. URL: https://www.bibcit.com/bibblog/en/image-conversion/equation-to-ms-word (date of access: 07.05.2026).

8. MathML editor. URL: https://math-editor.online/ (date of access: 07.05.2026).

9. CompSciLib. URL: https://www.compscilib.com/image-to-text (date of access: 07.05.2026).

10. Text2LaTex. URL: https://www.text2latex.com/ (date of access: 07.05.2026).

11. Matcha. URL: https://www.mathcha.io/ (date of access: 07.05.2026).

12. Learn C# Programming. URL: https://www.tutorialsteacher.com/csharp (date of access: 07.05.2026).

13. C# Tutorial. URL: https://www.w3schools.com/cs/index.php (date of access: 07.05.2026).

14. Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs (date of access: 07.05.2026).

15. Microsoft Visual Studio Tutorials. URL: https://visualstudiomagazine.com/pages/topic-pages/visual-studio-tutorials.aspx (date of access: 07.05.2026).

16. XML Tutorial. URL: https://www.w3schools.com/xml/ (date of access: 07.05.2026).

17. Extensible Markup Language. URL: https://www.w3.org/XML/ (date of access: 07.05.2026).

ДОДАТОК А

Технічне завдання

Вступ

Програмна система може використовуватися для перетворення математичних виразів.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Створення програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

А.2 Призначення розробки

Програмна система призначена для забезпечення процесу конвертації математичних конструкцій у кодове подання.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмна система для перетворення математичних конструкцій з текстових файлів у кодове подання забезпечує виконання таких функцій:

– підтримка можливості зчитування математичних конструкцій із текстових файлів та їх попереднього опрацювання для подальшого аналізу;

– підтримка можливості синтаксичного розбору математичних виразів із урахуванням операторів, функцій, змінних та вкладених структур;

– підтримка можливості формування внутрішнього представлення математичних конструкцій на основі XML-подібної моделі;

– підтримка можливості обробки спеціальних символів, зокрема літер грецького алфавіту та інших математичних позначень;

– можливість генерації програмного коду на основі структурованого представлення математичних виразів;

– підтримка можливості коректного врахування пріоритетів операцій та вкладеності виразів під час генерації коду;

– можливість відображення результатів перетворення у зручному для користувача вигляді в інтерфейсі програми.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмна система для перетворення математичних конструкцій з текстових файлів у кодове подання повинна забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання необхідна наявність персонального комп’ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання потрібні такі технічні та програмні ресурси: персональний комп’ютер або ноутбук з процесором, тактова частота якого не нижча за 2.0 ГГц, що дозволяє ефективно виконувати операції синтаксичного аналізу та генерації коду, обсяг оперативної пам’яті має становити не менше 4 ГБ. Для зберігання програмного забезпечення та оброблюваних даних необхідно не менше 500 МБ вільного місця на накопичувачі, при цьому використання твердотільного диска сприяє підвищенню швидкодії системи. Також передбачається наявність стандартних пристроїв введення та виведення інформації, таких як клавіатура, миша та дисплей.

З боку програмного забезпечення необхідною є наявність операційної системи сімейства Windows, починаючи з версії Windows 10 або новішої, що забезпечує підтримку сучасних засобів розробки та виконання програм. Обов’язковою умовою є встановлення платформи .NET відповідної версії, яка використовується для виконання програм, розроблених мовою C#. Для модифікації програмного продукту може знадобитися середовище розробки або виконання, таке як Visual Studio, що забезпечує підтримку необхідних бібліотек і компонентів. У випадку використання інтеграції з текстовим редактором додатково передбачається встановлення Microsoft Word, у якому реалізовано функціональність надбудови для роботи з математичними конструкціями.

А.3.6 Вимоги до маркування і пакування

Програмна система для перетворення математичних конструкцій з текстових файлів у кодове подання може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва – «Програмна система для перетворення математичних конструкцій з текстових файлів у кодове подання».

А.4 Вхідні дані до роботи

Вхідними даними до програмної системи є математичні конструкції (формули, рівняння) у текстових файлах.

Вихідними даними програмної системи є кодове подання відповідних математичних конструкцій.

ДОДАТОК Б

Фрагмент тексту програми

```
namespace MthCnvrtPrj
{
    public class MnFrm : Form
    {
void BtnPrss_Click(object sender, EventArgs e)
        {
            string txt = txtInpt.Text;
            XmlDocument xml = prsr.PrssTxt(txt);
            string cd = bldr.BldCd(xml);
            txtOtpt.Text = cd;
        }

        private void BtnLd_Click(object sender, EventArgs e)
        {
            OpenFileDialog dlg = new OpenFileDialog();
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                txtInpt.Text                                          =
System.IO.File.ReadAllText(dlg.FileName);
            }
        }

        private void BtnSv_Click(object sender, EventArgs e)
        {
            SaveFileDialog dlg = new SaveFileDialog();
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                System.IO.File.WriteAllText(dlg.FileName,
txtOtpt.Text);
            }
        }
    }

    public class PrsrMd
    {
        public XmlDocument PrssTxt(string txt)
        {
            XmlDocument doc = new XmlDocument();
            XmlElement root = doc.CreateElement("Expr");
            doc.AppendChild(root);

            List<string> tkns = Tknz(txt);
            int ndx = 0;
            XmlNode nd = PrssExpr(doc, tkns, ref ndx);
            root.AppendChild(nd);
```

```
            return doc;
        }

        private List<string> Tknz(string txt)
        {
            List<string> lst = new List<string>();
            StringBuilder bld = new StringBuilder();

            foreach (char c in txt)
            {
                if (char.IsLetterOrDigit(c))
                {
                    bld.Append(c);
                }
                else
                {
                    if (bld.Length > 0)
                    {
                        lst.Add(bld.ToString());
                        bld.Clear();
                    }
                    if (!char.IsWhiteSpace(c))
                    {
                        lst.Add(c.ToString());
                    }
                }
            }

            if (bld.Length > 0)
            {
                lst.Add(bld.ToString());
            }

            return lst;
        }

        private XmlNode PrssExpr(XmlDocument doc, List<string>
tkns, ref int ndx)
        {
            XmlElement op = doc.CreateElement("Op");
            op.SetAttribute("tp", "add");

            XmlNode lft = PrssTrm(doc, tkns, ref ndx);
            op.AppendChild(lft);
```

```
            while (ndx < tkns.Count && (tkns[ndx] == "+" ||
tkns[ndx] == "-"))
            {
                string sg = tkns[ndx++];
                XmlNode rgt = PrssTrm(doc, tkns, ref ndx);

                XmlElement nw = doc.CreateElement("Op");
                nw.SetAttribute("tp", sg == "+" ? "add" : "sub");
                nw.AppendChild(op);
                nw.AppendChild(rgt);
                op = nw;
            }

            return op;
        }

        private XmlNode PrssTrm(XmlDocument doc, List<string>
tkns, ref int ndx)
        {
            XmlElement op = doc.CreateElement("Op");
            op.SetAttribute("tp", "mul");

            XmlNode lft = PrssFct(doc, tkns, ref ndx);
            op.AppendChild(lft);

            while (ndx < tkns.Count && (tkns[ndx] == "*" ||
tkns[ndx] == "/"))
            {
                string sg = tkns[ndx++];
                XmlNode rgt = PrssFct(doc, tkns, ref ndx);

                XmlElement nw = doc.CreateElement("Op");
                nw.SetAttribute("tp", sg == "*" ? "mul" : "div");
                nw.AppendChild(op);
                nw.AppendChild(rgt);
                op = nw;
            }

            return op;
        }

        private XmlNode PrssFct(XmlDocument doc, List<string>
tkns, ref int ndx)
        {
            if (tkns[ndx] == "(")
            {
```

```
            ndx++;
            XmlNode nd = PrssExpr(doc, tkns, ref ndx);
            ndx++;
            return nd;
        }

        string tk = tkns[ndx++];
        XmlElement ndxEl;

        double nm;
        if (double.TryParse(tk, out nm))
        {
            ndxEl = doc.CreateElement("Cns");
            ndxEl.SetAttribute("vl", tk);
        }
        else
        {
            ndxEl = doc.CreateElement("Var");
            ndxEl.SetAttribute("nm", tk);
        }

        return ndxEl;
    }
}

public class BldCntxt
{
    public string BldCd(XmlDocument doc)
    {
        StringBuilder bld = new StringBuilder();
        XmlNode nd = doc.DocumentElement.FirstChild;
        string rs = BldNd(nd);
        bld.Append("double rslt = ");
        bld.Append(rs);
        bld.Append(";");
        return bld.ToString();
    }

    private string BldNd(XmlNode nd)
    {
        if (nd.Name == "Cns")
        {
            return nd.Attributes["vl"].Value;
        }

        if (nd.Name == "Var")
```

```
            {
                return nd.Attributes["nm"].Value;
            }

            if (nd.Name == "Op")
            {
                string tp = nd.Attributes["tp"].Value;
                string l = BldNd(nd.ChildNodes[0]);
                string r = BldNd(nd.ChildNodes[1]);

                if (tp == "add") return "(" + l + " + " + r +
")";
                if (tp == "sub") return "(" + l + " - " + r +
")";
                if (tp == "mul") return "(" + l + " * " + r +
")";
                if (tp == "div") return "(" + l + " / " + r +
")";
            }

            return "";
        }
    }

    static class PrgStrt
    {
        [STAThread]
        static void Mn()
        {
            Application.EnableVisualStyles();
            Application.Run(new MnFrm());
        }
    }
}

    public class UtlMd
    {
        private   Dictionary<string,   string>   mpng   =   new
Dictionary<string, string>();

        public UtlMd()
        {
            mpng["α"] = "alpha";
            mpng["β"] = "beta";
            mpng["γ"] = "gamma";
            mpng["δ"] = "delta";
```

```
    }

    public string NrmlzTxt(string txt)
    {
        StringBuilder bld = new StringBuilder();

        foreach (char c in txt)
        {
            string s = c.ToString();
            if (mpng.ContainsKey(s))
            {
                bld.Append(mpng[s]);
            }
            else
            {
                bld.Append(c);
            }
        }

        return bld.ToString();
    }

    public bool ChkNm(string txt)
    {
        if (string.IsNullOrEmpty(txt)) return false;

        foreach (char c in txt)
        {
            if (!char.IsLetterOrDigit(c)) return false;
        }

        return true;
    }
}

public class BldCntxtOptns
{
    public bool flgBrckt;
    public bool flgSpc;
    public bool flgUpr;

    public BldCntxtOptns()
    {
        flgBrckt = true;
        flgSpc = true;
        flgUpr = false;
```

```
    }

    public void StPrms(bool br, bool sp, bool up)
    {
        flgBrckt = br;
        flgSpc = sp;
        flgUpr = up;
    }

    public string FrmtOp(string l, string r, string op)
    {
        string sp = flgSpc ? " " : "";
        string rs = l + sp + op + sp + r;

        if (flgBrckt)
        {
            rs = "(" + rs + ")";
        }

        return rs;
    }

    public string NrmlzNm(string nm)
    {
        if (flgUpr)
        {
            return nm.ToUpper();
        }
        return nm.ToLower();
    }
}

public class AdvncdBldr
{
    private BldCntxtOptns optns;

    public AdvncdBldr()
    {
        optns = new BldCntxtOptns();
    }

    public string BldCmplx(XmlDocument doc)
    {
        StringBuilder bld = new StringBuilder();

        XmlNode nd = doc.DocumentElement.FirstChild;
```

```
        string rs = BldNdRcrs(nd);

        bld.Append("double fnl = ");
        bld.Append(rs);
        bld.Append(";");

        return bld.ToString();
    }

    private string BldNdRcrs(XmlNode nd)
    {
        if (nd.Name == "Pow")
        {
            string bs = BldNdRcrs(nd.ChildNodes[0]);
            string ex = BldNdRcrs(nd.ChildNodes[1]);
            return "Math.Pow(" + bs + ", " + ex + ")";
        }

        if (nd.Name == "Fnc")
        {
            string fn = nd.Attributes["nm"].Value;
            string arg = BldNdRcrs(nd.FirstChild);

            if (fn == "sin") return "Math.Sin(" + arg + ")";
            if (fn == "cos") return "Math.Cos(" + arg + ")";
            if (fn == "log") return "Math.Log(" + arg + ")";

            return arg;
        }

        if (nd.Name == "Var")
        {
            return optns.NrmlzNm(nd.Attributes["nm"].Value);
        }

        if (nd.Name == "Cns")
        {
            return nd.Attributes["vl"].Value;
        }

        if (nd.Name == "Op")
        {
            string tp = nd.Attributes["tp"].Value;
            string l = BldNdRcrs(nd.ChildNodes[0]);
            string r = BldNdRcrs(nd.ChildNodes[1]);
```

```
                if (tp == "add") return optns.FrmtOp(l, r, "+");
                if (tp == "sub") return optns.FrmtOp(l, r, "-");
                if (tp == "mul") return optns.FrmtOp(l, r, "*");
                if (tp == "div") return optns.FrmtOp(l, r, "/");
            }

            return "";
        }
    }

    public class ExprVldtr
    {
        public bool Vld(XmlDocument doc)
        {
            XmlNode nd = doc.DocumentElement;
            return ChkNd(nd);
        }

        private bool ChkNd(XmlNode nd)
        {
            if (nd == null) return false;

            foreach (XmlNode ch in nd.ChildNodes)
            {
                if (!ChkNd(ch)) return false;
            }

            if (nd.Name == "Op")
            {
                if (nd.ChildNodes.Count != 2) return false;
            }

            return true;
        }
    }

    public class XmlNrmlzr
    {
        public XmlDocument Nrmlz(XmlDocument doc)
        {
            XmlDocument nw = new XmlDocument();
            XmlElement rt = nw.CreateElement("Expr");
            nw.AppendChild(rt);

            XmlNode           nd           =           ClnNd(nw,
doc.DocumentElement.FirstChild);
```

```
        rt.AppendChild(nd);

        return nw;
    }

    private XmlNode ClnNd(XmlDocument doc, XmlNode nd)
    {
        XmlElement nw = doc.CreateElement(nd.Name);

        foreach (XmlAttribute at in nd.Attributes)
        {
            nw.SetAttribute(at.Name, at.Value);
        }

        foreach (XmlNode ch in nd.ChildNodes)
        {
            nw.AppendChild(ClnNd(doc, ch));
        }

        return nw;
    }
}

public class TxtPrprcsr
{
    public string ClnTxt(string txt)
    {
        StringBuilder bld = new StringBuilder();

        foreach (char c in txt)
        {
            if (!char.IsControl(c))
            {
                bld.Append(c);
            }
        }

        return bld.ToString();
    }

    public string RmvDblSpc(string txt)
    {
        while (txt.Contains("  "))
        {
            txt = txt.Replace("  ", " ");
        }
```

```
            return txt;
        }

        public string NrmlzLn(string txt)
        {
            return txt.Replace("\r", "").Replace("\n", " ");
        }
    }

    public class FnctnExtndr
    {
        public XmlNode BldFnc(XmlDocument doc, string nm, XmlNode
arg)
        {
            XmlElement el = doc.CreateElement("Fnc");
            el.SetAttribute("nm", nm);
            el.AppendChild(arg);
            return el;
        }

        public XmlNode BldPw(XmlDocument doc, XmlNode bs, XmlNode
ex)
        {
            XmlElement el = doc.CreateElement("Pow");
            el.AppendChild(bs);
            el.AppendChild(ex);
            return el;
        }

        public bool IsFnc(string tk)
        {
            return tk == "sin" || tk == "cos" || tk == "log";
        }
    }
}
```

ДОДАТОК В

Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Створення програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання

Виконав: Богдан ПОЧТАРЬ

ст. групи КНТ-212

Керівник: Сергій Олександрович СУББОТІН

д.т.н., професор, завідувач кафедри програмних засобів

Рисунок В.1 – Слайд 1

Об’єкт, предмет та мета роботи

- Об’єкт дослідження – процеси, пов’язані зі створенням програмних систем для перетворення даних.
- Предмет дослідження – програмні системи для конвертації даних.
- Метою роботи є розробка програмної системи перетворення математичних конструкцій з текстових файлів у кодове подання для зменшення витрат людських та часових ресурсів на обробку математичних даних.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

– виконати аналіз предметної області та програмних систем для перетворення даних;

– здійснити проєктування програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання;

– створити програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання;

– дослідити розроблену програмну систему для перетворення математичних конструкцій з текстових файлів у кодове подання.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Mathpix	Bibcit	MathML editor
Робота з текстовими файлами	+	+–	+
Підтримка складних математичних конструкцій	+	+–	+–
Інтеграція з текстовими редакторами	+–	+	+–
Редагування формул у реальному часі	–	–	+
Хмарна обробка	+	+	+–
Гнучкість налаштування формату виводу	+–	+	+–

4

Рисунок В.4 – Слайд 4

Постановка завдання

При розробці програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання необхідно забезпечити такі функціональні вимоги:
– підтримка можливості зчитування математичних конструкцій із текстових файлів та їх попереднього опрацювання для подальшого аналізу;
– підтримка можливості синтаксичного розбору математичних виразів із урахуванням операторів, функцій, змінних та вкладених структур;
– підтримка можливості формування внутрішнього представлення математичних конструкцій на основі XML-подібної моделі;
– підтримка можливості обробки спеціальних символів, зокрема літер грецького алфавіту та інших математичних позначень;
– можливість генерації програмного коду на основі структурованого представлення математичних виразів;
– підтримка можливості коректного врахування пріоритетів операцій та вкладеності виразів під час генерації коду;
– можливість відображення результатів перетворення у зручному для користувача вигляді в інтерфейсі програми.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Зручність роботи з текстом	+	+	+–
Строга типізація	+	–	+
Продуктивність	+	+–	+
Простота реалізації складної логіки	+	+	+–
Зручність налагодження	+	+	+–

6

Рисунок В.6 – Слайд 6

Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	SharpDevelop	MonoDevelop
Повнота функціоналу	+	–	+–
Продуктивність середовища	+	+–	+–
Підтримка тестування	+	–	+–
Зручність інтерфейсу	+	+	+–
Підтримка розширень	+	–	+–
Інтеграція з платформою .NET	+	+–	+–

7

Рисунок В.7 – Слайд 7

Вибір засобів зберігання та опрацювання даних

Критерій порівняння засобів зберігання даних	Засоби зберігання даних		
	XML	JSON	LaTeX
Ієрархічне представлення математичних конструкцій	+	+–	+–
Формалізованість і строгість структури	+	+–	–
Зручність автоматичного парсингу	+	+	–
Підтримка валідації структури	+	+–	+–
Зручність представлення складних формул	+	+–	+

8

Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

Рисунок В.10 – Слайд 10

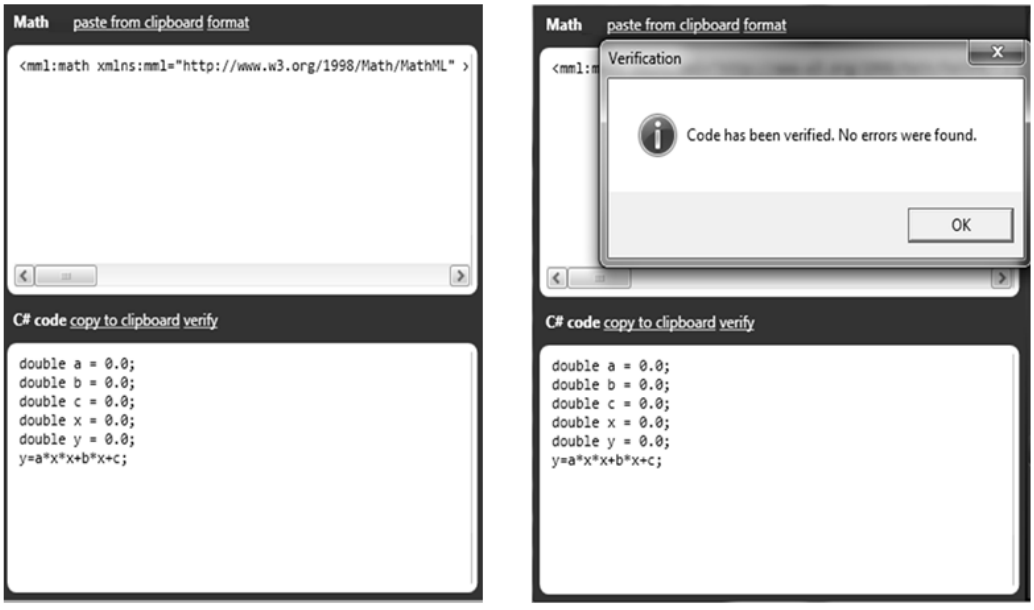
Подання математичного виразу за допомогою структури даних на основі XML

```
<Expression>
    <Operation type="add">
        <Operand>
            <Variable name="x"/>
        </Operand>
        <Operand>
            <Operation type="multiply">
                <Operand>
                    <Constant value="2"/>
                </Operand>
                <Operand>
                    <Variable name="y"/>
                </Operand>
            </Operation>
        </Operand>
    </Operation>
</Expression>
```

11

Рисунок В.11 – Слайд 11

Робота з програмою



12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процеси, пов'язані зі створенням програмних систем для перетворення даних.

Сформульовано функціональні вимоги до програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Для створення програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання обрано мову програмування C# та середовище розробки Visual Studio. Запропоновано структуру та описано основні модулі програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Описано функціонування програми. Виконано проєктування інтерфейсу взаємодії користувача з програмною системою для перетворення математичних конструкцій з текстових файлів у кодове подання.

Виконано тестування розробленої програмної системи для перетворення математичних конструкцій з текстових файлів у кодове подання. Результати тестування програми можна вважати позитивними, оскільки отримані результати підтверджують відповідність характеристик програми вимогам технічного завдання.

13

Рисунок В.13 – Слайд 13