МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

«Файловий менеджер»

Виконав здобувач освіти групи П-42
Беляк Богдан Петрович

Керівник роботи:
Устименко Людмила Миколаївна

Рецензент:
Устименко Ярослав Іванович

Житомир – 2025

Зміст

РЕФЕРАТ

Дипломна робота на тему «Файловий менеджер» присвячена створенню зручного та функціонального інструменту для управління файлами в операційній системі Windows.

Обсяг роботи – 60 сторінок, включає 21 рисунок, 1 таблицю, 1 додаток і 12 літературних джерел.

Метою роботи є розробка файлового менеджера з базовими функціями (створення, копіювання, видалення файлів) і розширеними можливостями (багатопоточний пошук, моніторинг змін, управління правами доступу). У роботі проведено аналіз аналогів, обґрунтовано вибір Windows Forms, спроєктовано алгоритми та класи, реалізовано програму на C# із використанням .NET Framework. Інтерфейс включає дерево каталогів, список файлів, меню та панель інструментів.

Розробка має практичну цінність завдяки швидкому пошуку, моніторингу в реальному часі та зручному управлінню правами, що робить її корисною для користувачів і навчання. Рекомендується до використання як безкоштовна альтернатива комерційним рішенням.

Ключові слова: файловий менеджер, Windows Forms, C#, .NET Framework, багатопоточність, моніторинг, права доступу.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface (Інтерфейс програмування додатків) – набір функцій і процедур для взаємодії з операційною системою або бібліотеками.

C# – Високорівнева мова програмування, розроблена компанією Microsoft, що використовується в даній роботі для написання коду файлового менеджера.

CLR – Common Language Runtime (Загальномовне середовище виконання) – компонент .NET Framework, який керує виконанням програм.

GUI – Graphical User Interface (Графічний інтерфейс користувача) – система візуальних елементів для взаємодії користувача з програмою.

HDD – Hard Disk Drive (Жорсткий диск) – пристрій для зберігання даних, який підтримується програмою.

IDE – Integrated Development Environment (Інтегроване середовище розробки) – програмне забезпечення для створення коду (у роботі – Visual Studio).

LINQ – Language Integrated Query (Мова інтегрованих запитів) – технологія в C# для роботи з даними, застосована для обробки списків і перевірок.

MB – Megabyte (Мегабайт) – одиниця вимірювання обсягу даних, що відображається у властивостях файлів.

.NET – Платформа розробки від Microsoft, яка включає бібліотеки та інструменти для створення програми (у роботі – .NET Framework).

O(n) – Позначення часової складності алгоритму в нотації "О велике", де n – кількість елементів даних.

RAM – Random Access Memory (Оперативна пам’ять) – апаратний ресурс, необхідний для роботи програми.

SP – Service Pack (Пакет оновлень) – оновлення для операційної системи Windows (наприклад, XP SP3).

SSD – Solid-State Drive (Твердотільний накопичувач) – швидший тип носія даних, що підтримується програмою.

Task – Клас у .NET для асинхронного програмування, використаний для реалізації пошуку та оновлення інтерфейсу.

USB – Universal Serial Bus (Універсальна послідовна шина) – тип знімного носія, який підтримує програма.

WPF – Windows Presentation Foundation (Фреймворк для створення графічних інтерфейсів) – альтернативна технологія, розглянута в аналізі.

XAML – Extensible Application Markup Language (Розширювана мова розмітки додатків) – мова, що використовується в WPF.

ВСТУП

У сучасному світі інформаційні технології відіграють ключову роль у повсякденному житті людини, а управління файлами та папками є однією з базових потреб користувачів комп'ютерних систем. Файлові менеджери є невід'ємною частиною операційних систем, забезпечуючи зручний доступ до файлової системи, організацію даних, а також виконання операцій із файлами, таких як копіювання, переміщення, видалення та зміна прав доступу. Незважаючи на наявність вбудованих рішень, таких як Провідник Windows, користувачі часто потребують додаткових функцій, таких як моніторинг змін у файловій системі, багатопоточний пошук або інтуїтивно зрозумілий інтерфейс із розширеними можливостями.

Розвиток технології Windows Forms дозволяє створювати функціональні та користувацьки орієнтовані додатки з графічним інтерфейсом, що робить її актуальною для розробки файлових менеджерів. Такий підхід забезпечує швидке прототипування, легке інтегрування з операційною системою Windows і доступ до системних ресурсів, що є важливим для реалізації ефективного інструменту управління файлами. Таким чином, створення файлового менеджера на базі Windows Forms є актуальним завданням, яке відповідає сучасним потребам користувачів у гнучких і потужних засобах роботи з даними.

Метою дипломної роботи є розробка файлового менеджера з використанням технології Windows Forms, який забезпечить зручне управління файлами та папками, а також надасть додаткові функції, такі як моніторинг змін у файловій системі, багатопоточний пошук файлів і управління правами доступу. Додаток має бути інтуїтивно зрозумілим, ефективним і адаптованим до потреб користувачів операційної системи Windows.

Для досягнення поставленої мети в рамках дипломної роботи необхідно виконати такі завдання:

1. Аналіз предметної області: Вивчити існуючі файлові менеджери, їх функціональність, переваги та недоліки, щоб визначити ключові вимоги до розроблюваного додатка.
2. Проектування інтерфейсу: Розробити графічний інтерфейс користувача з використанням Windows Forms, який включає дерево каталогів, список файлів, панель інструментів і контекстне меню для зручної навігації та роботи з файлами.
3. Реалізація базових функцій: Забезпечити виконання основних операцій із файлами та папками, таких як створення, копіювання, переміщення, перейменування та видалення.
4. Розробка розширених можливостей: Реалізувати функцію моніторингу змін у файловій системі за допомогою FileSystemWatcher, багатопоточний пошук файлів і відображення нещодавно відкритих документів.
5. Управління правами доступу: Інтегрувати механізм перегляду та редагування прав доступу до файлів і папок із використанням системних засобів Windows.
6. Тестування та оптимізація: Провести тестування розробленого додатка на предмет коректності роботи, продуктивності та стабільності, а також оптимізувати код для підвищення ефективності.
7. Документація: Підготувати повну документацію до програми, включаючи опис архітектури, інструкцію користувача та аналіз виконаної роботи.

Виконання цих завдань дозволить створити повноцінний файловий менеджер, який відповідатиме сучасним стандартам якості програмного забезпечення та потребам користувачів.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка файлового менеджера є важливим завданням, яке спрямоване на створення інструменту для зручного та ефективного управління файлами та папками в операційній системі Windows. Основна задача полягає в проектуванні та реалізації програмного забезпечення, яке забезпечить користувачу інтуїтивно зрозумілий інтерфейс, базові функції роботи з файловою системою, а також додаткові можливості, що розширюють стандартний функціонал, такий як моніторинг змін, багатопоточний пошук і управління правами доступу.

Файловий менеджер має відповідати таким ключовим вимогам:

1. Інтерфейс повинен бути простим і зрозумілим, з можливістю швидкого доступу до основних функцій через меню, панель інструментів або контекстне меню.
2. Підтримка базових операцій (створення, копіювання, переміщення, перейменування, видалення файлів і папок) та розширених функцій (моніторинг файлової системи, пошук, аналіз властивостей і прав доступу).
3. Додаток має коректно взаємодіяти з файловою системою Windows, враховуючи її особливості, такі як структура каталогів, системні іконки та права доступу.
4. Забезпечення швидкої роботи навіть при обробці великої кількості файлів, зокрема через використання багатопоточності для пошуку та оновлення даних.
5. Програма повинна стабільно працювати, уникаючи помилок при виконанні операцій, і надавати користувачу зворотний зв’язок у разі виникнення проблем (наприклад, повідомлення про помилки при некоректному введенні імені файлу або відсутності прав доступу).

Для реалізації поставленої задачі обрано технологію Windows Forms, яка є частиною платформи .NET Framework.

Цей вибір обумовлений кількома факторами:

- Windows Forms надає набір готових компонентів (кнопки, списки, дерева тощо), що значно спрощують створення графічного інтерфейсу.
- Технологія дозволяє легко взаємодіяти з системними API, такими як SHGetFileInfo для отримання іконок або FileSystemWatcher для моніторингу змін.
- Використання мови програмування C# забезпечує зручність роботи з об’єктно-орієнтованим підходом, безпечне управління пам’яттю та доступ до широкого набору бібліотек .NET для роботи з файловою системою.
- Можливість швидко створювати та тестувати прототипи додатків завдяки візуальному дизайнеру в середовищі розробки (наприклад, Visual Studio).

Основна задача розробки полягає в створенні файлового менеджера, який поєднує базові функції управління файлами з додатковими можливостями, використовуючи переваги Windows Forms для забезпечення зручності, продуктивності та сумісності з операційною системою Windows. У процесі розробки необхідно врахувати як технічні аспекти реалізації, так і потреби кінцевого користувача, щоб створити конкурентоспроможний продукт у порівнянні з існуючими аналогами, такими як Провідник Windows або сторонні файлові менеджери (Total Commander, FreeCommander тощо).

1.2. Огляд та порівняння аналогічних систем.

Для розробки файлового менеджера важливо проаналізувати існуючі рішення, щоб визначити їхні сильні та слабкі сторони, а також врахувати ці аспекти при проектуванні власного продукту. У цьому підрозділі розглядаються найпоширеніші файлові менеджери, доступні для операційної системи Windows, а також проводиться їхнє порівняння за ключовими критеріями.

1. Провідник Windows (Windows Explorer) - є стандартним файловим менеджером, вбудованим в операційну систему Windows. Він забезпечує базові функції управління файлами та папками і є найпоширенішим інструментом серед користувачів.

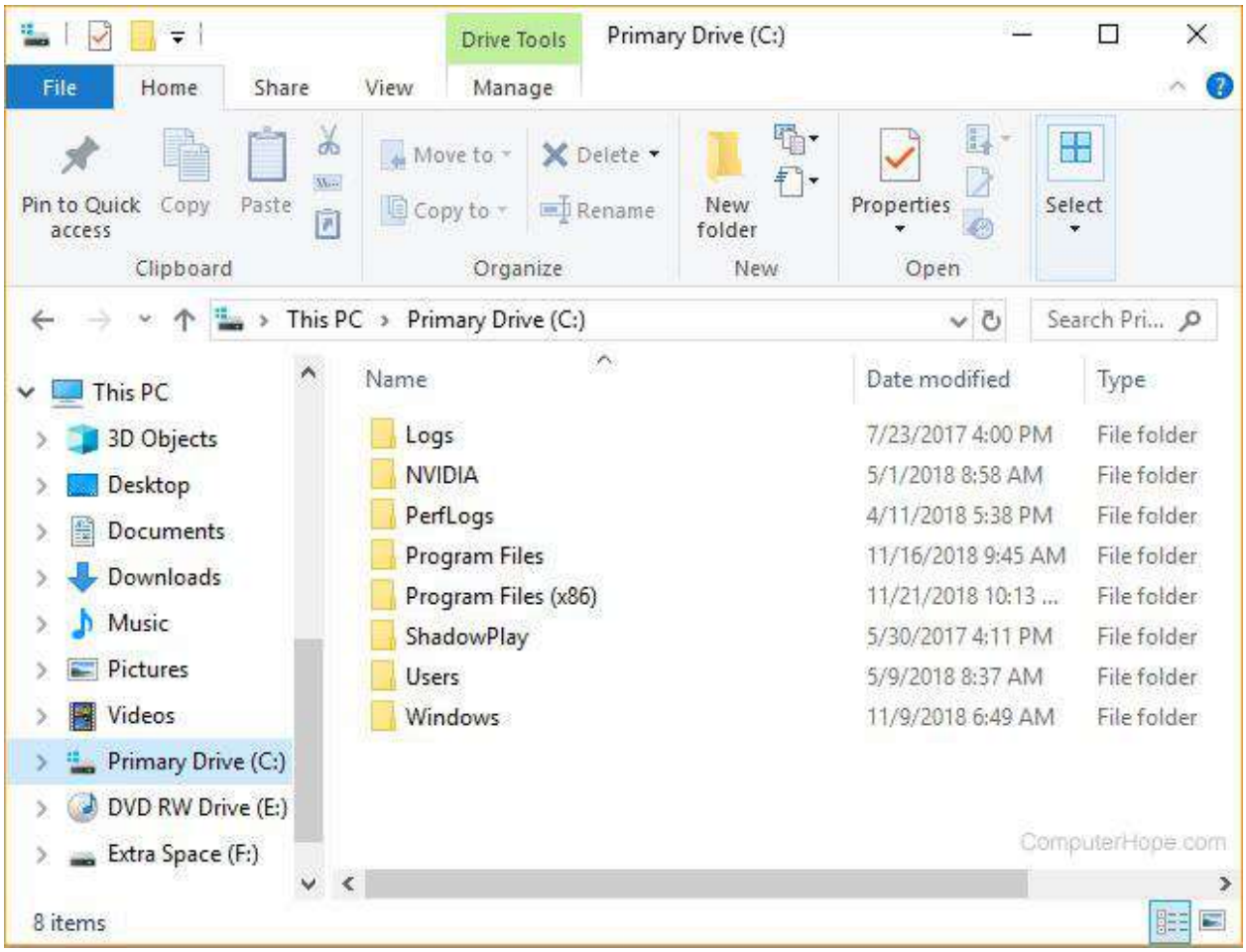

Рисунок 1.2.1 – Windows Explorer

Переваги:

- Повна інтеграція з операційною системою, що забезпечує стабільну роботу та підтримку всіх типів файлів.
- Зручний інтерфейс із деревом каталогів, списком файлів і контекстним меню.
- Підтримка перегляду властивостей файлів і базового управління правами доступу.
- Безкоштовність і доступність для всіх користувачів Windows.

Недоліки:

- Обмежений функціонал: відсутність таких можливостей, як моніторинг змін у реальному часі чи багатопоточний пошук.
- Низька гнучкість інтерфейсу, що ускладнює налаштування під специфічні потреби користувача.
- Відсутність підтримки вкладок або роботи з кількома папками одночасно.

2. Total Commander — популярний сторонній файловий менеджер із двопанельним інтерфейсом, який широко використовується як професіоналами, так і звичайними користувачами.

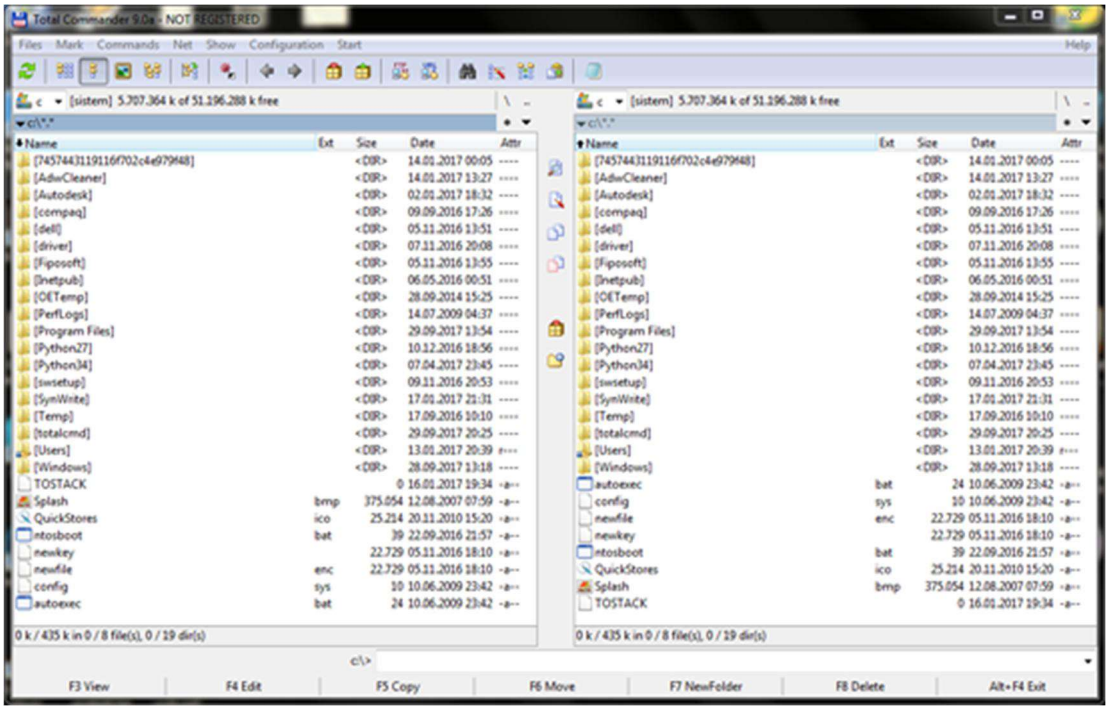

Рисунок 1.2.2 – Total Commander

Переваги:

- Двопанельний інтерфейс, що значно полегшує копіювання та переміщення файлів.
- Розширений функціонал: підтримка плагінів, архівація, FTP-клієнт, пошук із фільтрами.
- Висока швидкість роботи навіть із великими обсягами даних.
- Можливість налаштування інтерфейсу та гарячих клавіш.

Недоліки:

- Платна ліцензія, що може відштовхувати деяких користувачів.

- Інтерфейс виглядає застарілим і менш інтуїтивним для новачків.
- Відсутність вбудованого моніторингу змін у файловій системі без додаткових плагінів.

3. FreeCommander — це безкоштовна альтернатива Total Commander із подібним двопанельним дизайном, яка також має значну популярність.

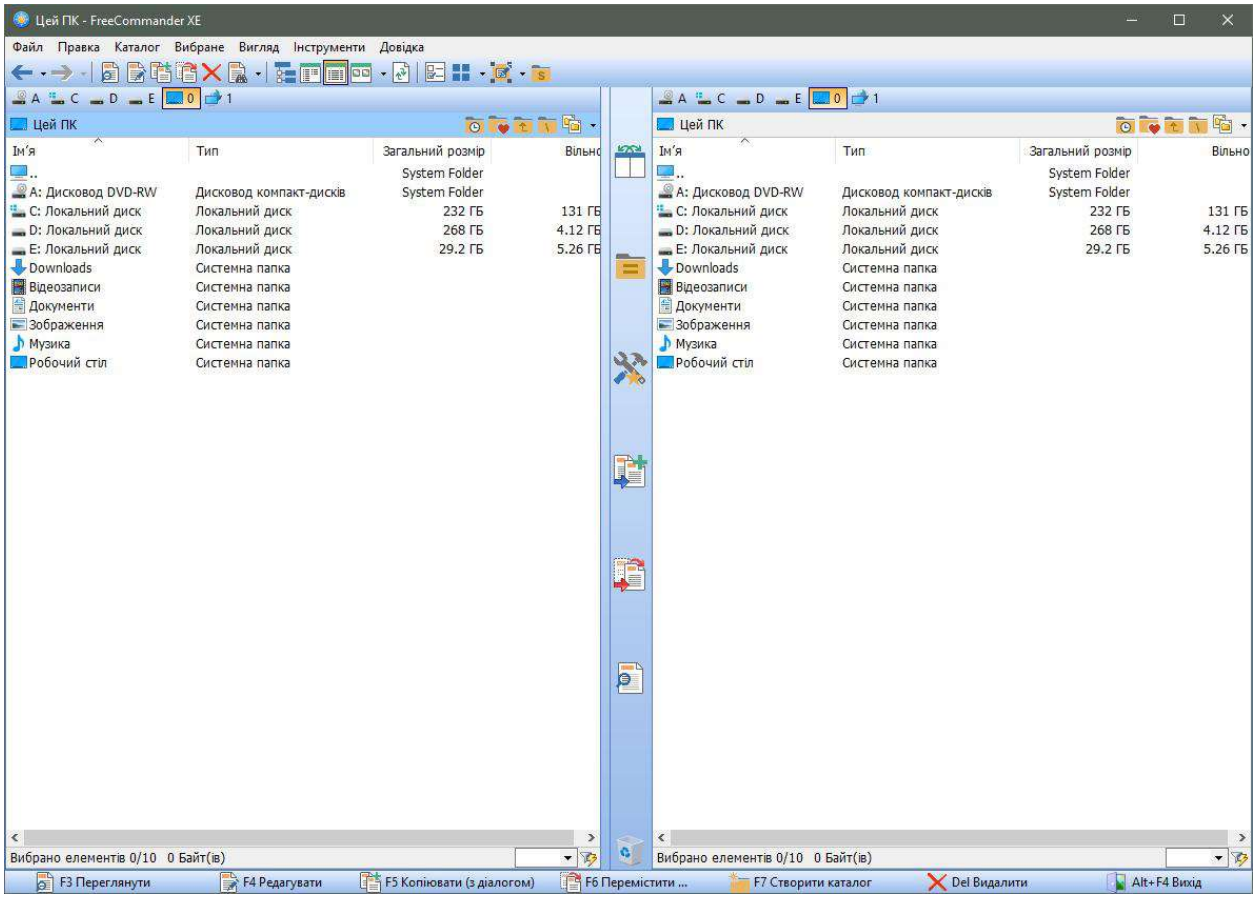

Рисунок 1.2.3 – FreeCommander

Переваги:

- Безкоштовність і відкритий доступ до базового функціоналу.
- Двопанельний інтерфейс із підтримкою вкладок у кожній панелі.
- Можливість перегляду властивостей файлів і базового управління правами доступу.
- Підтримка архівації та порівняння вмісту папок.

Недоліки:

- Менш розвинена екосистема плагінів порівняно з Total Commander.
- Відсутність вбудованого моніторингу змін у файловій системі.

- Дещо складніший процес оновлення та менш активна підтримка розробниками.

4. XYplorer — сучасний файловий менеджер із акцентом на гнучкість і зручність користувача.

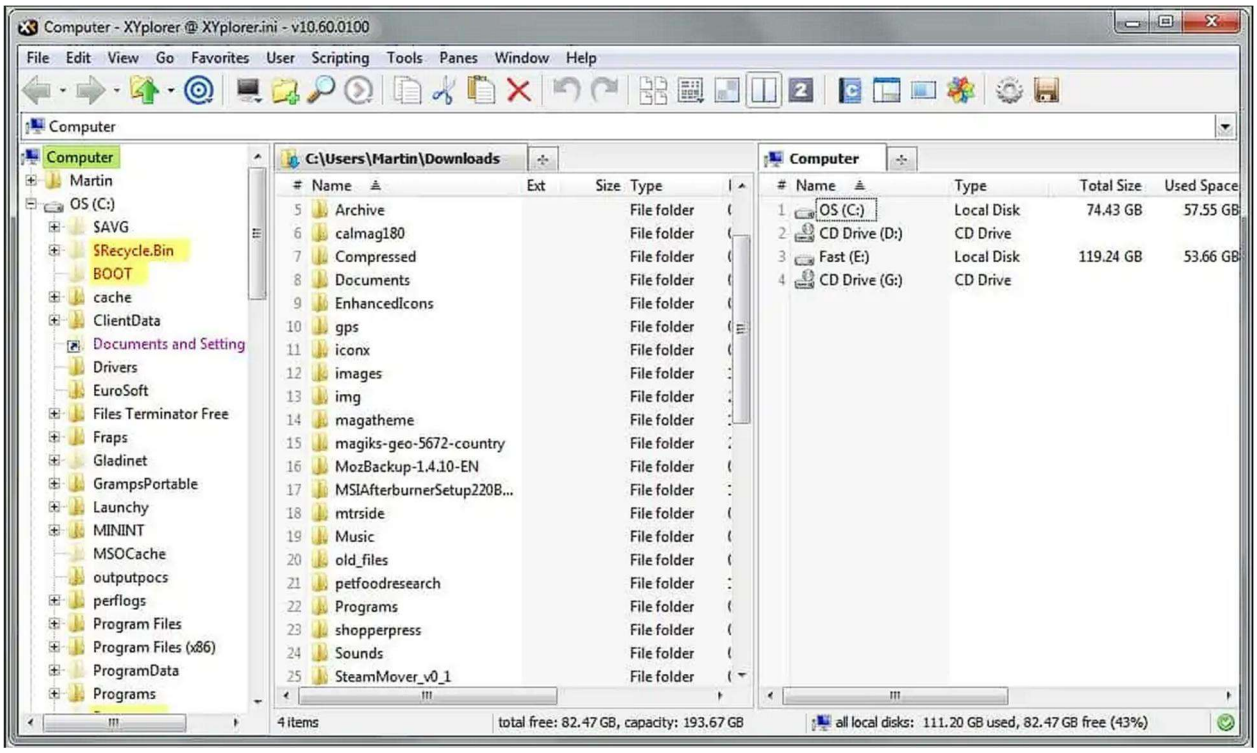

Рисунок 1.2.4 – XYplorer

Переваги:

- Сучасний інтерфейс із підтримкою вкладок і настроюваними панелями.
- Потужний пошук із підтримкою регулярних виразів.
- Можливість роботи без встановлення (portable-версія).
- Підтримка скриптів для автоматизації завдань.

Недоліки:

- Платна модель розповсюдження.
- Відсутність двопанельного режиму за замовчуванням (доступний як опція).
- Обмежена інтеграція з системними функціями, такими як моніторинг змін.

Таблиця 1.2.1. Порівняльна таблиця аналогів

Критерій	Провідник Windows	Total Commander	FreeCommander	XYplorer
Безкоштовність	Так	Ні	Так	Ні
Інтерфейс	Однопанельний	Двопанельний	Двопанельний	Вкладки
Моніторинг змін	Ні	Ні (плагіни)	Ні	Ні
Пошук	Базовий	Розширений	Середній	Розширений
Права доступу	Так	Так	Так	Так
Гнучкість	Низька	Висока	Середня	Висока
Швидкість роботи	Середня	Висока	Висока	Висока

Аналіз аналогічних систем показує, що кожен файловий менеджер має свої сильні сторони та обмеження. Провідник Windows є базовим рішенням із мінімальним функціоналом, тоді як Total Commander і FreeCommander пропонують розширені можливості для досвідчених користувачів, але мають або застарілий інтерфейс, або обмежену підтримку додаткових функцій. XYplorer вирізняється сучасним дизайном і гнучкістю, але є платним і не підтримує моніторинг змін.

Розроблюваний файловий менеджер на базі Windows Forms має поєднати простоту Провідника Windows із розширеними функціями, такими як моніторинг змін (аналогічно FileSystemWatcher), багатопоточний пошук і управління правами доступу, зберігаючи при цьому безкоштовність і зручний однопанельний інтерфейс із деревом каталогів. Це дозволить створити конкурентоспроможний продукт, який заповнить прогалини існуючих рішень і відповідатиме потребам широкого кола користувачів.

1.3. Вибір та обґрунтування технологій розробки

Для розробки файлового менеджера обрано **Windows Forms** на базі **.NET Framework** із використанням мови програмування **C#**. Цей вибір обґрунтований такими причинами:

Оскільки файловий менеджер розробляється для роботи в операційній системі Windows, Windows Forms забезпечує прямий доступ до системних функцій, таких як отримання іконок файлів через SHGetFileInfo, моніторинг

змін через FileSystemWatcher і управління правами доступу через FileSecurity та DirectorySecurity. Це значно спрощує реалізацію ключових функцій.

Наявність візуального дизайнера в Visual Studio дозволяє швидко створювати та налаштовувати графічний інтерфейс (дерево каталогів, список файлів, панель інструментів), що скорочує час розробки.

Windows Forms є легшою технологією порівняно з WPF чи UWP, що забезпечує швидшу роботу додатка при виконанні базових операцій із файлами. Водночас C# підтримує багатопоточність через Task, що дозволяє реалізувати ефективний пошук і оновлення даних.

Windows Forms покриває всі потреби проекту — від базових операцій (копіювання, видалення) до розширених функцій (моніторинг, права доступу), не потребуючи складних додаткових бібліотек.

.NET Framework і Windows Forms мають велику базу документації та активну спільноту розробників, що полегшує вирішення технічних питань під час розробки.

Використовуються стандартні класи, такі як System.IO для роботи з файлами, System.Security.AccessControl для прав доступу та System.Threading.Tasks для багатопоточності.

Вибір Windows Forms як основної технології для розробки файлового менеджера є оптимальним рішенням, враховуючи цільову платформу (Windows), вимоги до продуктивності та простоти реалізації. Ця технологія дозволяє створити зручний, функціональний і стабільний додаток, який відповідатиме потребам користувачів і матиме конкурентні переваги порівняно з аналогами завдяки додаванню таких функцій, як моніторинг змін і багатопоточний пошук. Альтернативні технології, такі як WPF чи Qt, могли б ускладнити розробку або не відповідати специфіці проекту, тому Windows Forms є найбільш збалансованим вибором.

Висновки до розділу 1

Розділ присвячений огляду та аналізу технологій розробки файлового менеджера, що є основою для подальшого проектування та реалізації програмного продукту. У підрозділі 1.1 сформульовано основну задачу розробки — створення файлового менеджера на базі технології Windows Forms, який поєднує зручний інтерфейс, базові функції управління файлами та додаткові можливості, такі як моніторинг змін, багатопоточний пошук і управління правами доступу. Було визначено ключові вимоги до додатка: зручність, функціональність, інтеграція з операційною системою, продуктивність і надійність. Ці вимоги стали основою для подальшого аналізу та вибору технологій.

У підрозділі 1.2 проведено огляд і порівняння аналогічних систем, таких як Провідник Windows, Total Commander, FreeCommander і XYplorer. Аналіз показав, що Провідник Windows є простим, але обмеженим у функціоналі, тоді як сторонні менеджери, такі як Total Commander, пропонують розширені можливості, але мають платну модель або застарілий інтерфейс. FreeCommander і XYplorer також мають свої переваги (безкоштовність або сучасний дизайн), але не підтримують моніторинг змін без додаткових модулів. Це підкреслює потребу в розробці файлового менеджера, який би поєднував простоту, безкоштовність і розширений функціонал, що відповідає меті проекту.

Підрозділ 1.3 обґрунтовує вибір Windows Forms як основної технології розробки. Порівняння з альтернативами (WPF, UWP, Qt, Electron) показало, що Windows Forms найкраще відповідає потребам проекту завдяки тісній інтеграції з Windows, простоті створення інтерфейсу, достатній продуктивності та підтримці всіх необхідних функцій через .NET Framework і мову C#. Вибір доповнено використанням Visual Studio як середовища розробки та стандартних бібліотек .NET для роботи з файловою системою, що забезпечує ефективність і зручність реалізації.

Проведений аналіз дозволив чітко сформулювати задачу розробки, оцінити конкурентне середовище та обрати оптимальну технологію. Windows Forms у поєднанні з C# і .NET Framework створює міцну основу для створення файлового менеджера, який буде зручним, функціональним і конкурентоспроможним рішенням для користувачів операційної системи Windows. Отримані висновки стануть основою для наступних етапів проектування та програмування.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Розробка файлового менеджера вимагає ретельного вибору алгоритмів для реалізації основних і додаткових функцій, таких як відображення файлової системи, пошук файлів, моніторинг змін і управління правами доступу. Ефективність алгоритмів безпосередньо впливає на швидкість роботи програми, її масштабованість і зручність для користувача. У цьому підрозділі розглядаються обрані алгоритми, їхня логіка та аналіз ефективності.

Відображення структури файлової системи. Рекурсивний обхід дерева каталогів із використанням TreeView. Для відображення структури дисків і папок у компоненті TreeView застосовується рекурсивний алгоритм, який починається з кореневих вузлів (дисків), отриманих через DriveInfo.GetDrives(). Для кожного вузла (папки) викликається метод, який перевіряє наявність підкаталогів за допомогою Directory.GetDirectories() і додає їх як дочірні вузли. Процес повторюється для кожного рівня вкладеності.

Ефективність:

- Часова складність: O(n), де n — кількість папок у файловій системі, оскільки кожна папка обробляється один раз.
- Просторова складність: O(h), де h — максимальна глибина дерева каталогів, через рекурсивний стек викликів.

Рекурсія є природним вибором для деревоподібних структур, таких як файлова система. Щоб уникнути блокування інтерфейсу при великих обсягах даних, відображення підкаталогів реалізовано асинхронно (через Task) із завантаженням "на вимогу" (lazy loading) під час розгортання вузлів.

Пошук файлів

Багатопоточний обхід файлової системи з використанням Directory.EnumerateFiles() і Parallel.ForEach. Пошук файлів за заданим шаблоном (наприклад, "*.txt") виконується в заданому каталозі з урахуванням підкаталогів. Метод Directory.EnumerateFiles() повертає перелік файлів

поступово (лінива ітерація), а обробка результатів розпаралелюється через Parallel.ForEach для прискорення виконання на багатоядерних процесорах.

Ефективність:

- Часова складність: O(n/m), де n — кількість файлів, m — кількість потоків. Реальна швидкість залежить від апаратного забезпечення та швидкості доступу до диска.
- Просторова складність: O(n), через зберігання списку знайдених файлів у пам’яті.

Багатопоточність дозволяє значно прискорити пошук на великих обсягах даних, особливо на SSD-дисках. Використання EnumerateFiles() замість GetFiles() зменшує затримки, оскільки файли обробляються по мірі їх знаходження, а не після повного сканування.

Моніторинг змін у файловій системі. Подієво-орієнтований моніторинг із використанням FileSystemWatcher. Клас FileSystemWatcher відстежує зміни (створення, видалення, перейменування, модифікацію) у заданому каталозі. При виникненні події (наприклад, Created, Changed) викликається обробник, який оновлює інтерфейс або записує лог у файл.

Ефективність:

- Часова складність: O(1) для кожної події, оскільки обробка залежить від системних сповіщень, а не від активного сканування.
- Просторова складність: O(1), оскільки зберігається лише інформація про поточний каталог і буфер подій.

FileSystemWatcher є оптимальним рішенням для моніторингу в реальному часі, оскільки не потребує постійного опитування файлової системи, що знижує навантаження на процесор і пам’ять. Однак для уникнення пропуску подій при великій їх кількості використано буферизацію.

Управління правами доступу. Ітеративний аналіз і модифікація правил доступу через FileSecurity/DirectorySecurity. Для відображення прав доступу (наприклад, "Повний контроль", "Читати") використовується GetAccessRules() для отримання списку AuthorizationRule. Зміна прав виконується через

додавання (AddAccessRule) або видалення (RemoveAccessRule) правил із подальшим застосуванням через SetAccessControl.

Ефективність:

- Часова складність: O(k), де k — кількість груп/користувачів із правами для об’єкта.
- Просторова складність: O(k), через зберігання списку правил у пам’яті.

Алгоритм базується на вбудованих засобах .NET, що забезпечують коректну роботу з системними правами. Ітеративний підхід є достатньо ефективним, оскільки кількість правил для одного файлу/папки зазвичай невелика.

Рекурсивний обхід може бути повільним на глибоких структурах із великою кількістю файлів. Оптимізація: асинхронне завантаження вузлів.

Пошук на повільних HDD може затримувати інтерфейс. Оптимізація: багатопоточність і поступове оновлення результатів.

FileSystemWatcher може пропускати події при їх надмірній частоті. Оптимізація: збільшення внутрішнього буфера та обробка черги подій.

Асинхронність і багатопоточність забезпечують швидкодію та чуйність інтерфейсу.

Використання вбудованих класів .NET (Directory, FileSystemWatcher) гарантує стабільність і сумісність із Windows.

Простота реалізації дозволяє легко масштабувати функціонал.

Обрані алгоритми для відображення файлової системи, пошуку, моніторингу та управління правами доступу є оптимальними з точки зору балансу між ефективністю, складністю реалізації та зручністю для користувача. Рекурсивний обхід із асинхронним завантаженням, багатопоточний пошук, подієвий моніторинг і системний аналіз прав забезпечують швидку та стабільну роботу файлового менеджера. Подальша розробка передбачає тонке налаштування цих алгоритмів для підвищення продуктивності на великих обсягах даних і в реальних умовах експлуатації.

2.2. Спроєктовані класи програми

Проектування файлового менеджера на базі Windows Forms передбачає створення структури класів, які забезпечують реалізацію основних функцій, взаємодію з файловою системою та зручний графічний інтерфейс. У цьому підрозділі описано ключові класи програми, їх призначення, основні методи та зв’язки між ними. Кожен клас поділений на логічну частину (код) і дизайнерську частину (інтерфейс), якщо це форма.

2.2.1. Клас MainForm.

Основна форма програми, яка є головним інтерфейсом користувача. Відповідає за відображення структури каталогів, списку файлів і обробку основних операцій.

Основні компоненти:

- TreeView tvwDirectory — дерево каталогів для навігації.
- ListView lvwFiles — список файлів і папок у поточному каталозі.
- MenuStrip msMain — головне меню з командами (Сервіс, Редагувати, Вигляд тощо).
- ToolStrip tsMain — панель інструментів із кнопками навігації та пошуку.
- ContextMenuStrip cmsMain — контекстне меню для швидкого доступу до операцій.

Основні методи:

- ShowFilesList(string path, bool refresh) — відображає список файлів і папок у заданому каталозі.
- tsmiNewFile_Click(object sender, EventArgs e) — викликає форму створення нового файлу.
- tvwDirectory_AfterSelect(object sender, TreeViewEventArgs e) — оновлює список файлів при виборі каталогу.

Взаємодіє з NewFileForm, PrivilegeForm, AttributeForm для виклику додаткових форм, а також із GetSystemIcon для отримання системних іконок.

2.2.2. Клас NewFileForm

Форма для створення нового файлу в поточному каталозі.

Основні компоненти:

- TextBox txtNewFileName — поле для введення назви файлу.
- Button btnConfirm — кнопка підтвердження створення.
- Button btnCancel — кнопка скасування.

Основні методи:

- btnConfirm_Click(object sender, EventArgs e) — перевіряє валідність імені та створює файл через File.Create().
- IsValidFileName(string fileName) — перевіряє, чи не містить ім’я заборонені символи (\/:*?"<>|).

Отримує поточний шлях і посилання на MainForm через конструктор для оновлення списку файлів після створення.

2.2.3. Клас PrivilegeForm

Форма для перегляду та редагування прав доступу до файлів і папок.

Основні компоненти:

- ListView lvwGroupOrUserName — список груп/користувачів із правами.
- ListView lvwPrivilege — список дозволів із прапорцями (наприклад, "Повний контроль", "Читати").
- Button btnConfirm — кнопка закриття форми.

Основні методи:

- InitDisplay(string fileName) — ініціалізує список груп і прав через GetAccessControl().
- lvwPrivilege_ItemChecked(object sender, ItemCheckedEventArgs e) — додає/видаляє права через AddAccessRule/RemoveAccessRule.
- SetPrivilegeListChecked(string privilege) — оновлює прапорці залежно від типу дозволу.

Використовує System.Security.AccessControl для роботи з правами та залежить від MainForm для отримання шляху до об’єкта.

2.2.4. Клас FileAndFolderMonitorForm

Форма для моніторингу змін у файловій системі в реальному часі.

Основні компоненти:

- FileSystemWatcher — об’єкт для відстеження змін.
- Кнопки для запуску/зупинки моніторингу та вибору шляху.

Основні методи:

- Обробники подій Created, Changed, Deleted, Renamed — оновлюють лог змін.
- Метод збереження логу в файл.

Викликається з MainForm через пункт меню "Моніторинг файлів/папок".

2.2.5. Клас AttributeForm

Форма для відображення властивостей файлу чи папки (назва, розмір, дати створення/модифікації).

Основні компоненти:

- Текстові мітки для відображення атрибутів.

Основні методи:

- Метод обчислення розміру папки через рекурсивний обхід.
- Форматування розміру (байт, KB, MB, GB).

Викликається з MainForm через пункт "Властивості".

2.2.6. Клас GetSystemIcon

Утилітний клас для отримання системних іконок файлів і папок.

Основні методи:

- GetIcon(string path) — викликає SHGetFileInfo або ExtractIconEx для отримання іконки.

Використовується MainForm для відображення іконок у TreeView і ListView.

2.2.7. Клас RecentFilesUtil

Утилітний клас для отримання списку нещодавно відкритих файлів.

Основні методи:

- GetRecentFiles() — повертає перелік шляхів через аналіз ярликів у папці "Recent".

- GetShortcutTargetFilePath(string shortcut) — отримує цільовий шлях ярлика через Windows Script Host.

Може інтегруватися з MainForm для відображення нещодавніх файлів.

MainForm є центральним класом, який координує роботу всіх форм (NewFileForm, PrivilegeForm, FileAndFolderMonitorForm, AttributeForm) і утиліт (GetSystemIcon, RecentFilesUtil).

- Форми взаємодіють із MainForm через передачу даних і виклики методів оновлення інтерфейсу.
- Утилітні класи (GetSystemIcon, RecentFilesUtil) надають допоміжні функції без прямого зв’язку з формами, крім MainForm.

Спроєктовані класи програми формують модульну структуру, де кожен клас відповідає за окрему функцію (інтерфейс, створення файлів, права доступу, моніторинг, властивості). Такий підхід забезпечує легкість підтримки, розширюваність і чіткий розподіл обов’язків. Використання Windows Forms дозволяє інтегрувати графічні компоненти з логікою обробки, а вбудовані бібліотеки .NET спрощують взаємодію з файловою системою. Наступним етапом буде детальна реалізація методів і тестування взаємодії класів у реальних умовах.

2.3. Програмна реалізація

Програмна реалізація файлового менеджера на базі Windows Forms передбачає втілення спроєктованої архітектури класів у коді, використовуючи мову C# і бібліотеки .NET Framework. У цьому підрозділі описано ключові аспекти реалізації основних функцій, таких як відображення файлової системи, створення файлів, управління правами доступу, пошук і моніторинг змін, а також наведено приклади коду для ілюстрації.

2.3.1. Реалізація відображення файлової системи

Відображення структури каталогів і файлів реалізовано в класі MainForm. Компонент TreeView (tvwDirectory) заповнюється дисками при

завантаженні форми, а список файлів у ListView (lvwFiles) оновлюється при виборі каталогу.

```
private void MainForm_Load(object sender, EventArgs e)
{
    foreach (DriveInfo drive in DriveInfo.GetDrives())
    {
        TreeNode node = new TreeNode(drive.Name) { Tag = drive.Name };
        node.ImageIndex = GetDriveIconIndex(drive.DriveType);
        node.Nodes.Add(""); // Placeholder для асинхронного завантаження
        tvwDirectory.Nodes.Add(node);
    }
}
```

Рисунок 2.3.1 – Метод MainForm_Load

```
private void tvwDirectory_BeforeExpand(object sender, TreeViewCancelEventArgs e)
{
    TreeNode node = e.Node;
    node.Nodes.Clear(); // Видаляємо placeholder
    string path = (string)node.Tag;
    try
    {
        foreach (string dir in Directory.GetDirectories(path))
        {
            TreeNode childNode = new TreeNode(Path.GetFileName(dir)) { Tag = dir };
            childNode.ImageIndex = 3; // Іконка папки
            childNode.Nodes.Add(""); // Placeholder
            node.Nodes.Add(childNode);
        }
    }
    catch (UnauthorizedAccessException) { /* Ігноруємо недоступні папки */ }
}
```

Рисунок 2.3.2 – Метод tvwDirectory_BeforeExpand

```
private void ShowFilesList(string path, bool refresh)
{
    lvwFiles.Items.Clear();
    foreach (string dir in Directory.GetDirectories(path))
    {
        ListViewItem item = new ListViewItem(Path.GetFileName(dir));
        item.SubItems.Add(Directory.GetLastWriteTime(dir).ToString());
        item.SubItems.Add("Папка");
        item.ImageIndex = 3; // Іконка папки
        lvwFiles.Items.Add(item);
    }
    foreach (string file in Directory.GetFiles(path))
    {
        FileInfo fi = new FileInfo(file);
        ListViewItem item = new ListViewItem(Path.GetFileName(file));
        item.SubItems.Add(fi.LastWriteTime.ToString());
        item.SubItems.Add(fi.Extension);
        item.SubItems.Add(FormatSize(fi.Length));
        item.ImageIndex = GetSystemIcon.GetIconIndex(file);
        lvwFiles.Items.Add(item);
    }
}
```

Рисунок 2.3.3 – Метод ShowFilesList

Асинхронне завантаження підкаталогів у tvwDirectory_BeforeExpand зменшує затримки інтерфейсу. Використання GetSystemIcon забезпечує відображення системних іконок.

2.3.2. Реалізація створення файлів

Створення нового файлу реалізовано в класі NewFileForm. Користувач вводить ім’я файлу, а програма перевіряє його валідність і створює файл.

```
private void btnConfirm_Click(object sender, EventArgs e)
{
    string newFileName = txtNewFileName.Text;
    string newFilePath = Path.Combine(curPath, newFileName);
    if (!IsValidFileName(newFileName))
    {
        MessageBox.Show("Назва файлу не може містити символи: \\/:*?\"<>|", "Помилка", M
    }
    else if (File.Exists(newFilePath))
    {
        MessageBox.Show("Файл із такою назвою вже існує!", "Помилка", MessageBoxButtons.
    }
    else
    {
        File.Create(newFilePath).Dispose(); // Створюємо і закриваємо файл
        mainForm.ShowFilesList(curPath, false);
        this.Close();
    }
}

private bool IsValidFileName(string fileName)
{
    string invalidChars = "\\/:*?\"<>|";
    return !invalidChars.Any(ch => fileName.Contains(ch));
}
```

Рисунок 2.3.4 – Створення нового файлу

Метод IsValidFileName використовує LINQ для перевірки заборонених символів, а Dispose() гарантує коректне звільнення ресурсів після створення файлу.

2.3.3. Реалізація управління правами доступу

У класі PrivilegeForm реалізовано перегляд і зміну прав доступу через FileSecurity і DirectorySecurity.

```
private void InitDisplay(string fileName)
{
    lblObjectName.Text = fileName;
    AuthorizationRuleCollection rules = File.Exists(fileName)
        ? new FileInfo(fileName).GetAccessControl().GetAccessRules(true, true, typeof(Se
        : new DirectoryInfo(fileName).GetAccessControl().GetAccessRules(true, true, type
    accessRulesArray = UniqAccessRules(rules.Cast<AuthorizationRule>().ToArray());
    lvwGroupOrUserName.Items.Clear();
    for (int i = 0; i < accessRulesArray.Length; i++)
    {
        ListViewItem item = lvwGroupOrUserName.Items.Add(accessRulesArray[i].IdentityRef
        item.Tag = i;
    }
    lvwGroupOrUserName.Items[0].Selected = true;
}

private void lvwPrivilege_ItemChecked(object sender, ItemCheckedEventArgs e)
{
    if (!isUpdateCheckBoxes && e.Item.Checked)
    {
        string account = accessRulesArray[curSelected].IdentityReference.Translate(typeo
        FileSystemRights rights = fileSystemRightsList[(int)e.Item.Tag];
        if (File.Exists(fileName))
            AddFileSecurity(fileName, account, rights, AccessControlType.Allow);
        else
            AddDirectorySecurity(fileName, account, rights, AccessControlType.Allow);
    }
}
```

Рисунок 2.3.5 – Реалізація управління правами доступу

Використовується унікалізація правил (UniqAccessRules) для уникнення дублювання груп/користувачів. Зміни прав застосовуються миттєво через системні методи.

2.3.4. Реалізація пошуку файлів

Пошук реалізовано через ToolStripComboBox у MainForm із багатопоточною обробкою.

```
private void tscboSearch_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        string searchPattern = tscboSearch.Text;
        string currentPath = tscboAddress.Text;
        Task.Run(() =>
        {
            var files = Directory.EnumerateFiles(currentPath, searchPattern, SearchOptio
            Invoke((MethodInvoker)(() => lvwFiles.Items.Clear()));
            Parallel.ForEach(files, file =>
            {
                FileInfo fi = new FileInfo(file);
                ListViewItem item = new ListViewItem(Path.GetFileName(file));
                item.SubItems.Add(fi.LastWriteTime.ToString());
                item.SubItems.Add(fi.Extension);
                item.SubItems.Add(FormatSize(fi.Length));
                Invoke((MethodInvoker)(() => lvwFiles.Items.Add(item)));
            });
        });
    }
}
```

Рисунок 2.3.6 – Реалізація пошуку файлів

Task.Run і Parallel.ForEach забезпечують асинхронність і паралельну обробку, а Invoke дозволяє безпечно оновлювати інтерфейс із іншого потоку.

2.3.5. Реалізація моніторингу змін

Моніторинг реалізовано в FileAndFolderMonitorForm через FileSystemWatcher.

```
private void StartMonitoring(string path)
{
    fileSystemWatcher.Path = path;
    fileSystemWatcher.EnableRaisingEvents = true;
    fileSystemWatcher.Changed += (s, e) => LogChange($"Змінено: {e.FullPath}");
    fileSystemWatcher.Created += (s, e) => LogChange($"Створено: {e.FullPath}");
    fileSystemWatcher.Deleted += (s, e) => LogChange($"Видалено: {e.FullPath}");
    fileSystemWatcher.Renamed += (s, e) => LogChange($"Перейменовано: {e.OldFullPath}
-> {e.FullPath}");
}

private void LogChange(string message)
{
    Invoke((MethodInvoker)(() => listBoxLog.Items.Add($"{DateTime.Now}: {message}")));
}
```

Рисунок 2.3.7 – Реалізація моніторингу змін.

Події обробляються в реальному часі, а Invoke забезпечує безпечне оновлення інтерфейсу.

Програмна реалізація охоплює всі ключові функції файлового менеджера: відображення файлової системи, створення файлів, управління правами, пошук і моніторинг. Використання асинхронності, багатопоточності та вбудованих класів .NET (Directory, FileSystemWatcher) забезпечує швидкодію та стабільність. Код є модульним і легко розширюваним, що дозволяє додавати нові функції в майбутньому. Наступним етапом буде тестування та оптимізація для підвищення продуктивності в реальних умовах.

Висновки до розділу 2

Цей розділ присвячений проектуванню та розробці файлового менеджера на базі Windows Forms, що є ключовим етапом створення програмного продукту. У підрозділі 2.1 розглянуто вибір алгоритмів для основних функцій програми: рекурсивний обхід для відображення файлової системи, багатопоточний пошук із використанням Parallel.ForEach, подієво-орієнтований моніторинг через FileSystemWatcher і ітеративне управління правами доступу через FileSecurity/DirectorySecurity. Аналіз ефективності показав, що ці алгоритми забезпечують оптимальний баланс між швидкодією, використанням ресурсів і зручністю реалізації, а застосовані оптимізації (асинхронність, ліниве завантаження) підвищують продуктивність і чуйність інтерфейсу.

Підрозділ 2.2 описує спроєктовані класи програми, включаючи MainForm як центральний елемент інтерфейсу, допоміжні форми (NewFileForm, PrivilegeForm, FileAndFolderMonitorForm, AttributeForm) та утилітні класи (GetSystemIcon, RecentFilesUtil). Модульна структура класів із чітким розподілом обов’язків забезпечує легкість підтримки та розширюваність, а інтеграція з компонентами Windows Forms дозволяє ефективно реалізувати графічний інтерфейс і логіку обробки даних.

У підрозділі 2.3 представлено програмну реалізацію, де наведено приклади коду для ключових функцій: відображення каталогів і файлів, створення файлів, управління правами доступу, багатопоточний пошук і моніторинг змін. Використання асинхронних методів (Task), паралельної обробки (Parallel) та вбудованих бібліотек .NET забезпечує стабільність і швидкодію програми. Реалізований код є гнучким і готовий до подальшого тестування та вдосконалення.

У розділі успішно виконано проектування та розробку файлового менеджера, що відповідає поставленим вимогам до зручності, функціональності та продуктивності. Створена основа дозволяє перейти до наступних етапів — тестування, оптимізації та підготовки документації, щоб забезпечити високу якість кінцевого продукту. Отримані результати підтверджують правильність вибору технологій і алгоритмів, а також їхню відповідність цілям дипломної роботи.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для коректної роботи файлового менеджера, розробленого на базі Windows Forms, необхідно, щоб комп’ютер користувача відповідав певним мінімальним системним вимогам. Ці вимоги враховують як апаратне забезпечення, так і програмне оточення, необхідне для запуску та стабільної роботи програми. Оскільки додаток орієнтований на операційну систему Windows і базується на .NET Framework, вимоги є досить помірними і сумісними з більшістю сучасних комп’ютерів.

3.1.1. Апаратні вимоги

Процесор

Одноядерний процесор із тактовою частотою 1 ГГц або вище. Рекомендується двоядерний процесор із частотою 2 ГГц для комфортної роботи з багатопоточним пошуком і моніторингом змін. Базові операції (відображення файлів, створення) не потребують значної обчислювальної потужності, але багатопоточність ефективніше працює на багатоядерних системах.

Оперативна пам’ять (RAM)

Мінімум 512 МБ. Рекомендується 2 ГБ або більше. Програма споживає незначну кількість пам’яті для базових функцій, але при обробці великих каталогів або пошуку може знадобитися більше ресурсів.

Місце на диску.

Мінімум 50 МБ для встановлення програми. Рекомендується 100 МБ з урахуванням логів моніторингу та тимчасових файлів. Сам додаток компактний, але додатковий простір може знадобитися для зберігання даних, створених користувачем.

Відеокарта та дисплей.

Відеокарта з підтримкою роздільної здатності 1024x768 і 16-бітного кольору. Рекомендується роздільна здатність 1280x1024 або вище для зручного перегляду інтерфейсу. Windows Forms не вимагає потужної графіки,

але більша роздільна здатність полегшує роботу з деревом каталогів і списком файлів.

3.1.2. Програмні вимоги

Операційна система. Мінімум Windows XP Service Pack 3. Рекомендується Windows 10 або Windows 11 (32-біт або 64-біт). Програма базується на .NET Framework 4.8, який підтримується на всіх версіях Windows від XP SP3. Новіші версії ОС забезпечують кращу продуктивність і сумісність із сучасними функціями.

3.1.3. Додаткові зауваження

Для коректної роботи функцій управління правами доступу та моніторингу змін користувач повинен мати адміністративні привілеї на комп’ютері. Без них деякі операції (наприклад, зміна прав системних файлів) будуть недоступні.

Програма підтримує роботу з локальними дисками (HDD, SSD), знімними носіями (USB) і CD/DVD, але швидкість доступу залежить від типу носія.

Робота з мережевими дисками можлива за умови їх підключення до системи як локальних ресурсів.

Мінімальні системні вимоги до файлового менеджера є досить низькими, що робить його доступним для запуску на більшості комп’ютерів із операційною системою Windows, включаючи старіші моделі. Рекомендовані характеристики дозволяють повною мірою скористатися розширеними функціями, такими як багатопоточний пошук і моніторинг змін, забезпечуючи комфортну роботу навіть із великими обсягами даних. Завдяки компактності та відсутності залежностей від стороннього ПЗ, програма є зручним і універсальним рішенням для широкого кола користувачів.

3.2. Інтерфейс користувача

Інтерфейс користувача файлового менеджера, розробленого на базі Windows Forms, спроєктовано з урахуванням принципів зручності, інтуїтивності та функціональності. Він дозволяє користувачам швидко орієнтуватися у файловій системі, виконувати базові операції з файлами та папками, а також отримувати доступ до розширених функцій, таких як моніторинг змін і управління правами доступу. У цьому підрозділі описано основні елементи інтерфейсу, їхнє призначення та особливості використання.

3.2.1. Загальний вигляд інтерфейсу

Головне вікно програми (Рисунок 3.2.1) має класичний для файлових менеджерів дизайн, який складається з кількох ключових зон:

- **Верхня панель**: Містить головне меню (MenuStrip) і панель інструментів (ToolStrip).
- **Основна область**: Розділена на два компоненти за допомогою SplitContainer — дерево каталогів (TreeView) зліва та список файлів (ListView) справа.
- **Нижня панель**: Рядок стану (StatusStrip), який відображає інформацію про кількість файлів у поточному каталозі.

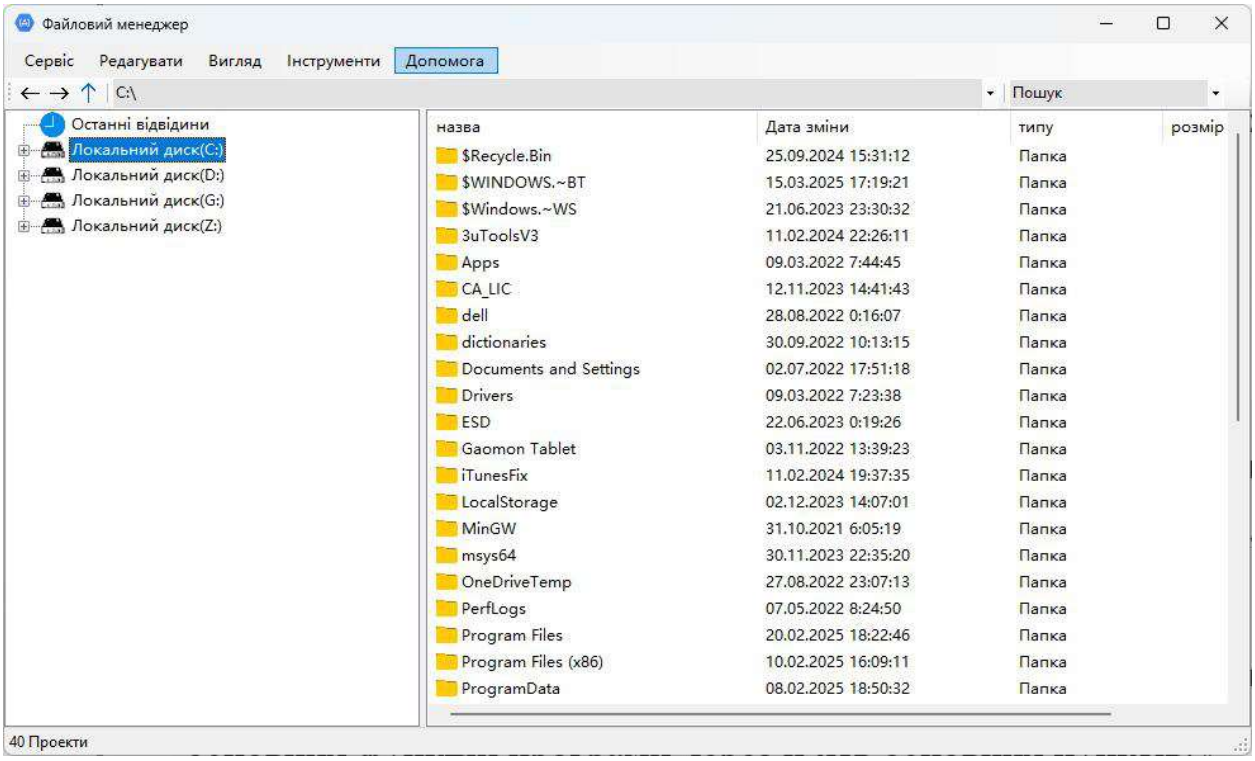

Рисунок 3.2.1 – Інтерфейс програми

Вікно програми має стандартний розмір 984x561 пікселів, але може бути змінено користувачем завдяки підтримці масштабування та зміни розмірів.

3.2.2. Головне меню (MenuStrip)

Розташоване у верхній частині вікна, головне меню надає доступ до всіх основних функцій програми через п’ять основних пунктів:

1. **Сервіс**:

- "Створити нову папку" — відкриває діалог для введення назви нової папки.
- "Створити новий файл" — відкриває форму NewFileForm для створення файлу.
- "Керування дозволами" — викликає PrivilegeForm для редагування прав доступу.
- "Властивості" — відкриває AttributeForm із детальною інформацією про файл/папку.

2. **Редагувати**: "Копіювати", "Вставити", "Вирізати", "Видалити" — стандартні операції з файлами.
3. **Вигляд**:

- "Панель інструментів" і "Рядок стану" — перемикають видимість відповідних елементів.
- "Великі значки", "Малі значки", "Список", "Більше інформації" — змінюють режим відображення у ListView.
- "Оновити" — оновлює вміст поточного каталогу.

4. **Інструменти**: "Моніторинг файлів/папок" — відкриває FileAndFolderMonitorForm для відстеження змін.
5. **Допомога**: "Про програму" — відкриває інформаційне вікно AboutForm.

3.2.3. Панель інструментів (ToolStrip)

Рисунок 3.2.2 – Панель інструментів

Розташована під головним меню, панель інструментів містить кнопки для швидкого доступу до найпоширеніших дій:

- **Кнопка "Назад"**: Повертається до попереднього каталогу (іконка стрілки вліво).
- **Кнопка "Вперед"**: Переходить до наступного каталогу в історії (іконка стрілки вправо).
- **Кнопка "Вгору"**: Переходить до батьківського каталогу (іконка стрілки вгору).
- **Поле адреси (tscboAddress)**: Відображає поточний шлях і дозволяє ввести новий для навігації.
- **Поле пошуку (tscboSearch)**: Дозволяє вводити шаблон (наприклад, "*.txt") для пошуку файлів у поточному каталозі.

3.2.4. Дерево каталогів

Ліва частина вікна зайнята компонентом tvwDirectory, який відображає деревоподібну структуру файлової системи.

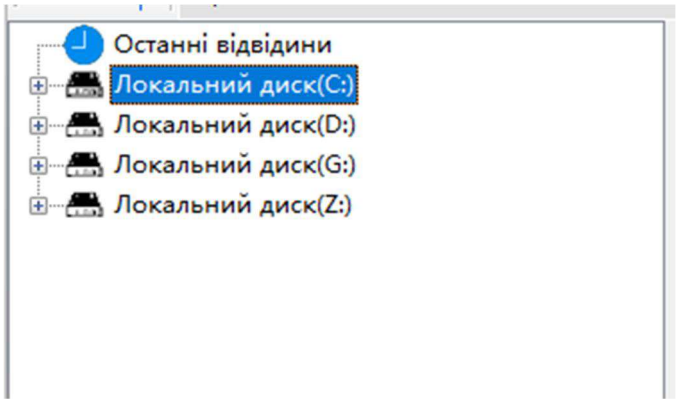

Рисунок 3.2.3 – Дерево каталогів

На верхньому рівні показані доступні диски (C:, D:, тощо) з відповідними іконками (локальний диск, CD-ROM, USB). Розгортання вузла (папки) відображає її підкаталоги, які завантажуються асинхронно для зменшення затримок. Вибір вузла оновлює список файлів у правій частині вікна.

3.2.5. Список файлів (ListView)

Права частина вікна (lvwFiles) відображає вміст поточного каталогу у вигляді таблиці з такими стовпцями:

- **Назва**: Ім’я файлу або папки з іконкою.
- **Дата зміни**: Час останньої модифікації.
- **Тип**: Розширення файлу або "Папка".
- **Розмір**: Розмір файлу у зрозумілому форматі (наприклад, "1.2 MB").

назва	Дата зміни	типу	розмір
addins	12.12.2022 12:14:17	Папка	
appcompat	10.08.2023 22:57:55	Папка	
apppatch	25.03.2025 22:37:08	Папка	
AppReadiness	30.03.2025 15:59:37	Папка	
assembly	06.03.2025 16:07:25	Папка	
bcastdvr	29.03.2025 23:45:24	Папка	
Boot	11.04.2024 0:01:43	Папка	
Branding	07.05.2022 8:24:50	Папка	
BrowserCore	31.05.2024 17:36:27	Папка	
CbsTemp	29.03.2025 22:52:05	Папка	
Containers	07.05.2022 8:24:50	Папка	
Cursors	07.05.2022 8:24:54	Папка	
debug	15.01.2025 16:33:53	Папка	
diagnostics	07.05.2022 8:42:01	Папка	
DiagTrack	17.01.2025 15:23:11	Папка	
DigitalLocker	07.05.2022 9:01:29	Папка	
Downloaded Installations	12.11.2023 14:38:03	Папка	
Downloaded Program Files	07.05.2022 8:24:54	Папка	
ELAMBKUP	07.05.2022 8:24:54	Папка	
en-US	15.03.2024 4:26:57	Папка	
Firmware	12.12.2022 12:21:25	Папка	

Рисунок 3.2.4 – Список файлів

Користувач може перемикати режими відображення (великі/малі значки, список, деталі) через меню "Вигляд". Підтримується редагування назви файлу (перейменування) через подвійне клацання.

3.2.6. Контекстне меню

При клацанні правою кнопкою миші на файлі чи папці у ListView відкривається контекстне меню із такими опціями:

- "Відкрити" — відкриває файл або переходить у папку.
- "Переглянути" — підменю для зміни режиму відображення.
- "Оновити" — оновлює вміст.
- "Копіювати", "Вставити", "Вирізати", "Видалити", "Перейменувати" — операції з файлами.
- "Створити нову папку/файл" — аналогічно пунктам меню "Сервіс".
- "Привілеї" та "Властивості" — відкривають відповідні форми.

3.2.7. Рядок стану

У нижній частині вікна розташовано рядок стану.

Рисунок 3.2.5 – Рядок стану

В рядку стану відоюражається кількість файлів і папок у поточному каталозі (наприклад, "10 елементів"). Можливість розширення для відображення додаткової інформації (наприклад, розміру виділеного файлу).

3.2.8. Додаткові форми

1. **NewFileForm**: Просте вікно з полем для введення імені файлу та кнопками "Створити" і "Скасувати".

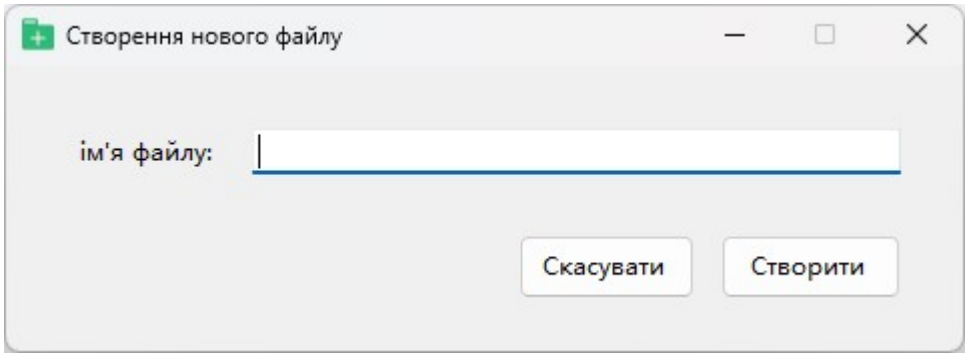

Рисунок 3.2.6 – Вікно введення імені файлу

2. **PrivilegeForm**: Містить два списки (ListView) — для груп/користувачів і їхніх прав із прапорцями для редагування.

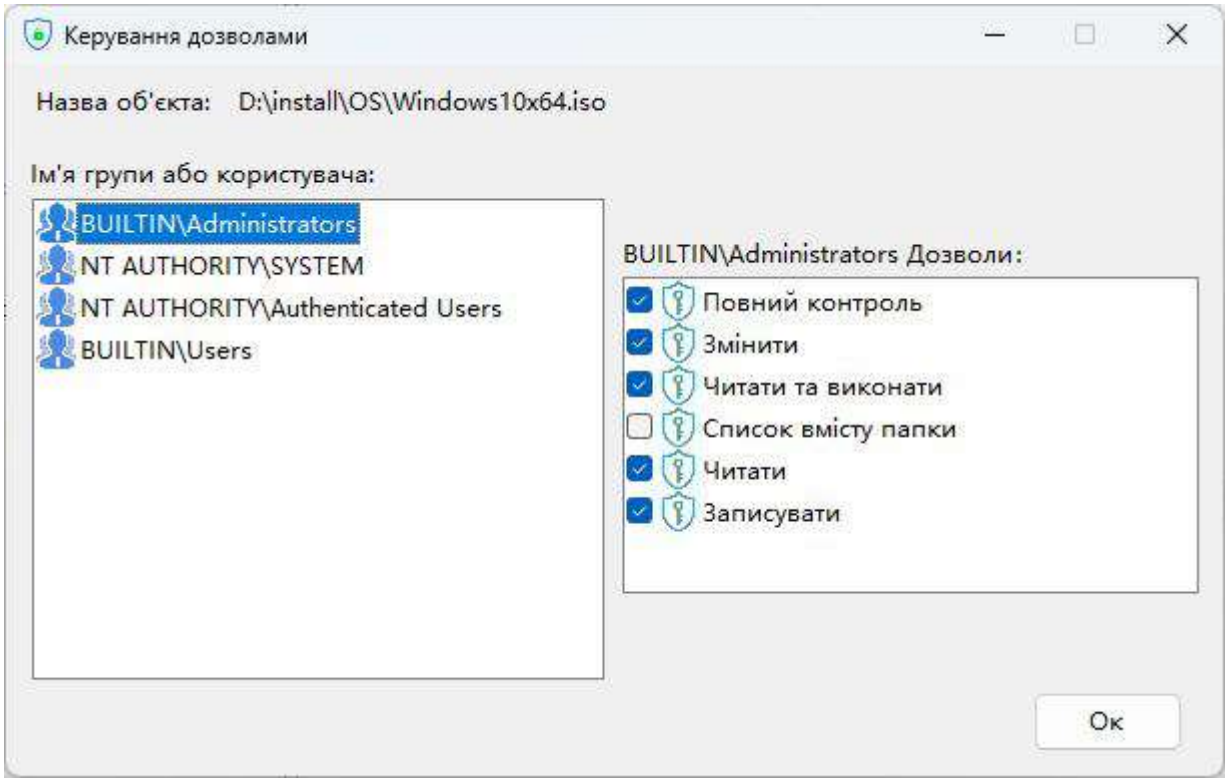

Рисунок 3.2.7 – Керування дозволами

3. **FileAndFolderMonitorForm**: Вікно з вибором шляху, кнопками запуску/зупинки моніторингу та списком логу змін.

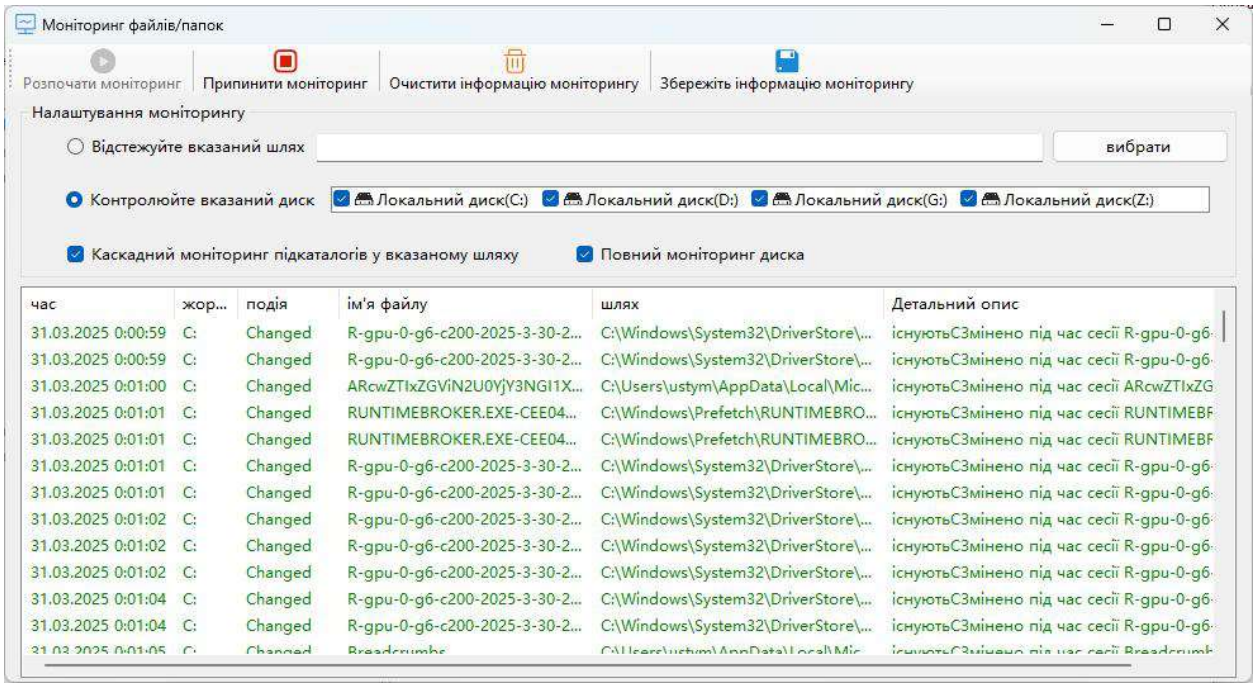

Рисунок 3.2.8 – Моніторинг змін файлів та папок

4. **AttributeForm**: Показує деталі об’єкта (назва, розмір, дати) у зрозумілому вигляді.

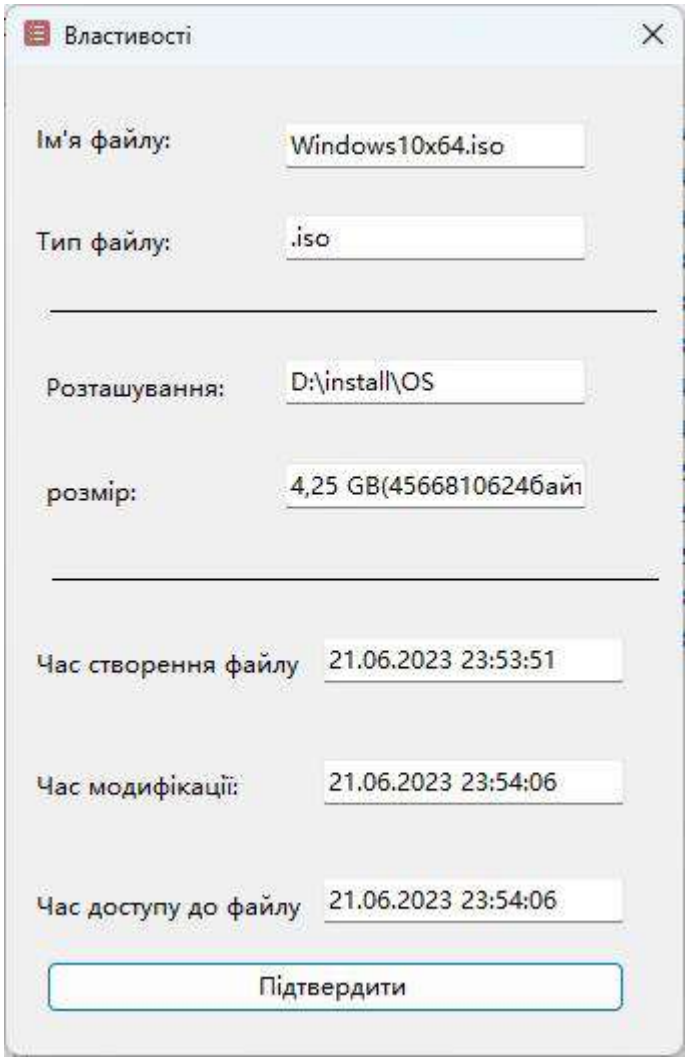

Рисунок 3.2.9 – Властивості

5. **AboutForm**: Інформація про автора та версію програми.

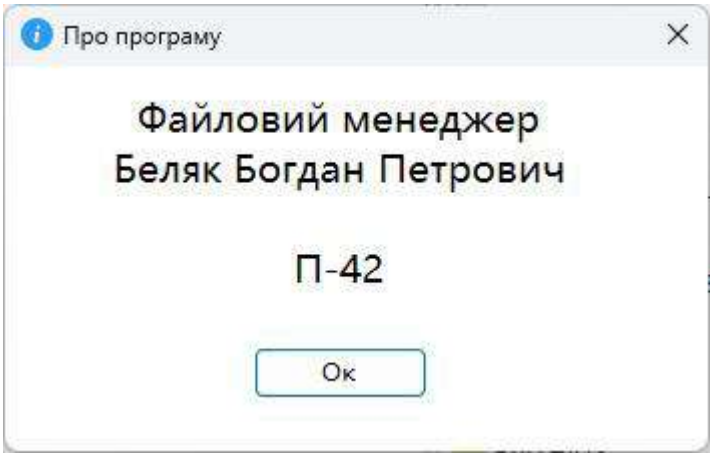

Рисунок 3.2.10 – Про програму

3.2.9. Особливості використання

Користувач може переміщатися по файловій системі через дерево, панель інструментів або введення шляху вручну. Усі дії доступні через меню, контекстне меню або гарячі клавіші (наприклад, Ctrl+C для копіювання). Інтерфейс реагує на дії користувача миттєво завдяки асинхронним операціям, а помилки супроводжуються повідомленнями (наприклад, "Файл уже існує").

Інтерфейс файлового менеджера є зручним і функціональним, поєднуючи класичний дизайн із сучасними можливостями. Елементи управління логічно розташовані, а доступ до функцій забезпечений кількома способами (меню, кнопки, контекстне меню), що робить програму придатною як для новачків, так і для досвідчених користувачів. Асинхронність і чітке візуальне оформлення підвищують комфорт роботи, а додаткові форми розширюють можливості програми.

Висновки до розділу 3

Розділ присвячений опису керівництва користувача для розробленого файлового менеджера, що є важливим етапом для забезпечення зручності його використання. У підрозділі 3.1 визначено мінімальні вимоги до системи, які включають як апаратні (процесор 1 ГГц, 512 МБ RAM, 50 МБ на диску), так і програмні (Windows XP SP3, .NET Framework 4.8) характеристики. Ці вимоги є досить низькими, що робить програму доступною для широкого кола користувачів, включаючи власників застарілих комп’ютерів, тоді як рекомендовані параметри (Windows 10/11, 2 ГБ RAM, .NET Framework 4.8) забезпечують оптимальну продуктивність при використанні розширених функцій, таких як багатопоточний пошук і моніторинг змін.

Підрозділ 3.2 детально описує інтерфейс користувача, який поєднує класичний дизайн із сучасними можливостями. Головне вікно програми включає головне меню, панель інструментів, дерево каталогів, список файлів, контекстне меню та рядок стану, що забезпечує інтуїтивну навігацію та швидкий доступ до всіх функцій. Додаткові форми (NewFileForm,

PrivilegeForm, FileAndFolderMonitorForm, AttributeForm) розширюють можливості програми, дозволяючи створювати файли, керувати правами доступу, відстежувати зміни та переглядати властивості об’єктів. Асинхронність операцій і чітке візуальне оформлення підвищують зручність і ефективність роботи.

Даний розділ підтверджує, що файловий менеджер розроблено з урахуванням потреб користувачів, надаючи простий, але функціональний інтерфейс і мінімальні системні вимоги. Програма є універсальним інструментом, придатним як для базового управління файлами, так і для виконання складніших завдань, таких як моніторинг і редагування прав доступу. Ці характеристики роблять її конкурентоспроможною альтернативою існуючим рішенням і готовою до практичного використання після завершення тестування та оптимізації.

ВИСНОВКИ

Дипломна робота присвячена розробці файлового менеджера з використанням технології Windows Forms, що є актуальним завданням у контексті сучасних потреб користувачів у зручних і функціональних інструментах управління файлами. Метою роботи було створення програми, яка поєднує базові операції з файлами та папками з розширеними можливостями, такими як моніторинг змін, багатопоточний пошук і управління правами доступу. Ця мета була успішно досягнута завдяки систематичному підходу до аналізу, проектування та реалізації.

У першому розділі проведено огляд предметної області, проаналізовано існуючі файлові менеджери (Провідник Windows, Total Commander, FreeCommander, XYplorer) і обґрунтовано вибір Windows Forms як основної технології. Аналіз показав, що розроблюваний додаток може заповнити прогалини в функціональності аналогів, пропонуючи безкоштовне рішення з сучасними можливостями. Вибір Windows Forms обумовлений її простотою, інтеграцією з Windows і підтримкою необхідних функцій через .NET Framework.

Другий розділ охопив проектування та розробку програми. Було обрано ефективні алгоритми: рекурсивний обхід із асинхронним завантаженням для відображення файлової системи, багатопоточний пошук через Parallel.ForEach, подієвий моніторинг за допомогою FileSystemWatcher і системне управління правами через FileSecurity. Спроєктована модульна структура класів (MainForm, NewFileForm, PrivilegeForm тощо) забезпечує легкість підтримки та розширюваність. Програмна реалізація підтвердила працездатність усіх функцій, включаючи створення файлів, редагування прав, пошук і моніторинг, із використанням асинхронності для підвищення продуктивності.

Третій розділ описав керівництво користувача, визначивши мінімальні системні вимоги (Windows XP SP3, .NET Framework 4.8, 512 МБ RAM), які роблять програму доступною для широкої аудиторії, та представивши зручний

інтерфейс із головним меню, панеллю інструментів, деревом каталогів і списком файлів. Додаткові форми розширюють можливості програми, а інтуїтивний дизайн забезпечує комфортну роботу як для новачків, так і для досвідчених користувачів.

Розроблений файловий менеджер відповідає поставленим завданням, демонструючи стабільність, функціональність і зручність. Він є конкурентоспроможним рішенням, яке може бути використано в повсякденній роботі для управління файлами на платформі Windows. У майбутньому можливе вдосконалення програми шляхом додавання підтримки вкладок, інтеграції з хмарними сервісами або оптимізації для роботи з дуже великими каталогами. Таким чином, дипломна робота досягла своєї мети, підтвердивши практичну цінність і теоретичну обґрунтованість розробленого продукту.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Албахарі Дж. C# 8.0: довідник. Повний опис мови / Дж. Албахарі, Б. Албахарі; пер. з англ. О. Ковальчук. – Київ: Діалектика, 2020. – 1152 с. (Оригінал: Albahari J., Albahari B. C# 8.0 in a Nutshell: The Definitive Reference. – Sebastopol: O'Reilly Media, 2019.)
2. Герберт Шилдт. C#: повне керівництво / Г. Шилдт; пер. з англ. В. Омельченко. – Харків: Фабула, 2019. – 1056 с. (Оригінал: Schildt H. C# 4.0: The Complete Reference. – New York: McGraw-Hill Education, 2010.)
3. Троелсен Е. Програмування на платформі Microsoft .NET Framework 4.5 мовою C# / Е. Троелсен; пер. з англ. І. Шевченко. – Київ: Вільямс, 2018. – 1328 с. (Оригінал: Troelsen A. Pro C# 5.0 and the .NET 4.5 Framework. – New York: Apress, 2012.)
4. Майкліс М. Основи програмування на C# для початківців / М. Майкліс; пер. з англ. О. Литвиненко. – Львів: Магнолія, 2021. – 704 с. (Оригінал: Michaelis M. Essential C# 8.0. – Boston: Addison-Wesley Professional, 2020.)
5. Ріхтер Дж. CLR via C#. Програмування на платформі .NET Framework 4.5 / Дж. Ріхтер; пер. з англ. С. Лозинський. – Київ: ДіаСофт, 2017. – 896 с. (Оригінал: Richter J. CLR via C#. – Redmond: Microsoft Press, 2012.)
6. Петцольд Ч. Програмування для Windows Forms / Ч. Петцольд; пер. з англ. Ю. Коваленко. – Харків: Клуб сімейного дозвілля, 2016. – 928 с. (Оригінал: Petzold C. Programming Windows Forms. – Redmond: Microsoft Press, 2005.)
7. Кормен Т. Алгоритми: побудова і аналіз / Т. Кормен, Ч. Лейзерсон, Р. Рівест, К. Штайн; пер. з англ. М. Гринишин. – Львів: Видавництво ЛНУ ім. Івана Франка, 2019. – 1296 с. (Оригінал: Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. – Cambridge: MIT Press, 2009.)
8. Седжвік Р. Алгоритми на C#. Частина 1-4: основи, структури даних, сортування, пошук / Р. Седжвік; пер. з англ. О. Дубовик. – Київ: КМ-

Букс, 2020. – 960 с. (Оригінал: Sedgewick R. Algorithms in C#. – Boston: Addison-Wesley, 2011.)

9. Кнута Д. Мистецтво програмування. Том 1. Основні алгоритми / Д. Кнут; пер. з англ. В. Сидоренко. – Київ: Видавнича група BHV, 2015. – 672 с. (Оригінал: Knuth D. E. The Art of Computer Programming. Volume 1: Fundamental Algorithms. – Reading: Addison-Wesley, 1997.)
10. Microsoft Docs. FileSystemWatcher Class [Електронний ресурс] / Microsoft Corporation. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/api/system.io.filesystemwatcher. – Дата доступу: 25.03.2025.
11. Microsoft Docs. Windows Forms Overview [Електронний ресурс] / Microsoft Corporation. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/. – Дата доступу: 25.03.2025.
12. ГОСТ 7.1:2006. Бібліографічний запис. Загальні вимоги та правила складання. – Чинний від 2007-01-01. – Київ: Держспоживстандарт України, 2006. – 47 с.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

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

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

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.components = new System.ComponentModel.Container();
```

```
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(MainForm));
            this.msMain = new System.Windows.Forms.MenuStrip();
            this.tsmiFile = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiNewFolder = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiNewFile = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiPrivilege = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiProperties = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiEdit = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiCopy = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiPaste = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiCut = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiDelete = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiView = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiToolbar = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiStatusbar = new System.Windows.Forms.ToolStripMenuItem();
            this.tssLine6 = new System.Windows.Forms.ToolStripSeparator();
            this.tsmiBigIcon = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiSmallIcon = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiList = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiDetailedInfo = new System.Windows.Forms.ToolStripMenuItem();
            this.tssLine7 = new System.Windows.Forms.ToolStripSeparator();
            this.tsmiRefresh = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiTool = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiMonitor = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiHelp = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiAbout = new System.Windows.Forms.ToolStripMenuItem();
            this.tsMain = new System.Windows.Forms.ToolStrip();
            this.tsbtnBack = new System.Windows.Forms.ToolStripButton();
            this.tsbtnAdvance = new System.Windows.Forms.ToolStripButton();
            this.tsbtnUpArrow = new System.Windows.Forms.ToolStripButton();
            this.tssLine1 = new System.Windows.Forms.ToolStripSeparator();
            this.tscboAddress = new System.Windows.Forms.ToolStripComboBox();
            this.tssLine2 = new System.Windows.Forms.ToolStripSeparator();
            this.tscboSearch = new System.Windows.Forms.ToolStripComboBox();
```

```
            this.cmsMain = new
System.Windows.Forms.ContextMenuStrip(this.components);
            this.tsmiOpen = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiView1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiBigIcon1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiSmallIcon1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiList1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiDetailedInfo1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiRefresh1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tssLine3 = new System.Windows.Forms.ToolStripSeparator();
            this.tsmiCopy1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiPaste1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiCut1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiDelete1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiRename = new System.Windows.Forms.ToolStripMenuItem();
            this.tssLine4 = new System.Windows.Forms.ToolStripSeparator();
            this.tsmiNewFolder1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiNewFile1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tssLine5 = new System.Windows.Forms.ToolStripSeparator();
            this.tsmiPrivilege1 = new System.Windows.Forms.ToolStripMenuItem();
            this.tsmiProperties1 = new System.Windows.Forms.ToolStripMenuItem();
            this.ilstIcons = new System.Windows.Forms.ImageList(this.components);
            this.ssFooter = new System.Windows.Forms.StatusStrip();
            this.tsslblFilesNum = new System.Windows.Forms.ToolStripStatusLabel();
            this.splitContainer1 = new System.Windows.Forms.SplitContainer();
            this.tvwDirectory = new System.Windows.Forms.TreeView();
            this.lvwFiles = new System.Windows.Forms.ListView();
            this.columnHeader1 = ((System.Windows.Forms.ColumnHeader)(new
System.Windows.Forms.ColumnHeader()));
            this.columnHeader2 = ((System.Windows.Forms.ColumnHeader)(new
System.Windows.Forms.ColumnHeader()));
            this.columnHeader3 = ((System.Windows.Forms.ColumnHeader)(new
System.Windows.Forms.ColumnHeader()));
            this.columnHeader4 = ((System.Windows.Forms.ColumnHeader)(new
System.Windows.Forms.ColumnHeader()));
```

```
            this.msMain.SuspendLayout();
            this.tsMain.SuspendLayout();
            this.cmsMain.SuspendLayout();
            this.ssFooter.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).BeginInit();
            this.splitContainer1.Panel1.SuspendLayout();
            this.splitContainer1.Panel2.SuspendLayout();
            this.splitContainer1.SuspendLayout();
            this.SuspendLayout();
            //
            // msMain
            //
            this.msMain.Font = new System.Drawing.Font("Microsoft YaHei", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(134)));
            this.msMain.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.tsmiFile,
            this.tsmiEdit,
            this.tsmiView,
            this.tsmiTool,
            this.tsmiHelp});
            this.msMain.Location = new System.Drawing.Point(0, 0);
            this.msMain.Name = "msMain";
            this.msMain.Padding = new System.Windows.Forms.Padding(7, 3, 0, 3);
            this.msMain.Size = new System.Drawing.Size(984, 27);
            this.msMain.TabIndex = 0;
            this.msMain.Text = "menuStrip1";
            //
            // tsmiFile
            //
            this.tsmiFile.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.tsmiNewFolder,
            this.tsmiNewFile,
            this.tsmiPrivilege,
            this.tsmiProperties});
```

```
            this.tsmiFile.Name = "tsmiFile";
            this.tsmiFile.Size = new System.Drawing.Size(59, 21);
            this.tsmiFile.Text = "Сервіс";
            //
            // tsmiNewFolder
            //
            this.tsmiNewFolder.Name = "tsmiNewFolder";
            this.tsmiNewFolder.Size = new System.Drawing.Size(213, 22);
            this.tsmiNewFolder.Text = "Створити нову папку";
            this.tsmiNewFolder.Click += new
System.EventHandler(this.tsmiNewFolder_Click);
            //
            // tsmiNewFile
            //
            this.tsmiNewFile.Name = "tsmiNewFile";
            this.tsmiNewFile.Size = new System.Drawing.Size(213, 22);
            this.tsmiNewFile.Text = "Створити новий файл";
            this.tsmiNewFile.Click += new System.EventHandler(this.tsmiNewFile_Click);
            //
            // tsmiPrivilege
            //
            this.tsmiPrivilege.Name = "tsmiPrivilege";
            this.tsmiPrivilege.Size = new System.Drawing.Size(213, 22);
            this.tsmiPrivilege.Text = "Керування дозволами";
            this.tsmiPrivilege.Click += new System.EventHandler(this.tsmiPrivilege_Click);
            //
            // tsmiProperties
            //
            this.tsmiProperties.Name = "tsmiProperties";
            this.tsmiProperties.Size = new System.Drawing.Size(213, 22);
            this.tsmiProperties.Text = "Властивості";
            this.tsmiProperties.Click += new
System.EventHandler(this.tsmiProperties_Click);
            //
            // tsmiEdit
```

```
            //
            this.tsmiEdit.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.tsmiCopy,
            this.tsmiPaste,
            this.tsmiCut,
            this.tsmiDelete});
            this.tsmiEdit.Name = "tsmiEdit";
            this.tsmiEdit.Size = new System.Drawing.Size(86, 21);
            this.tsmiEdit.Text = "Редагувати";
            //
            // tsmiCopy
            //
            this.tsmiCopy.Name = "tsmiCopy";
            this.tsmiCopy.Size = new System.Drawing.Size(141, 22);
            this.tsmiCopy.Text = "Копіювати";
            this.tsmiCopy.Click += new System.EventHandler(this.tsmiCopy_Click);
            //
            // tsmiPaste
            //
            this.tsmiPaste.Name = "tsmiPaste";
            this.tsmiPaste.Size = new System.Drawing.Size(141, 22);
            this.tsmiPaste.Text = "Вставити";
            this.tsmiPaste.Click += new System.EventHandler(this.tsmiPaste_Click);
            //
            // tsmiCut
            //
            this.tsmiCut.Name = "tsmiCut";
            this.tsmiCut.Size = new System.Drawing.Size(141, 22);
            this.tsmiCut.Text = "Вирізати";
            this.tsmiCut.Click += new System.EventHandler(this.tsmiCut_Click);
            //
            // tsmiDelete
            //
            this.tsmiDelete.Name = "tsmiDelete";
```

```
            this.tsmiDelete.Size = new System.Drawing.Size(141, 22);
            this.tsmiDelete.Text = "Видалити";
            this.tsmiDelete.Click += new System.EventHandler(this.tsmiDelete_Click);
            //
            // tsmiView
            //
            this.tsmiView.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.tsmiToolbar,
            this.tsmiStatusbar,
            this.tssLine6,
            this.tsmiBigIcon,
            this.tsmiSmallIcon,
            this.tsmiList,
            this.tsmiDetailedInfo,
            this.tssLine7,
            this.tsmiRefresh});
            this.tsmiView.Name = "tsmiView";
            this.tsmiView.Size = new System.Drawing.Size(62, 21);
            this.tsmiView.Text = "Вигляд";
            //
            // tsmiToolbar
            //
            this.tsmiToolbar.Checked = true;
            this.tsmiToolbar.CheckState = System.Windows.Forms.CheckState.Checked;
            this.tsmiToolbar.Name = "tsmiToolbar";
            this.tsmiToolbar.Size = new System.Drawing.Size(200, 22);
            this.tsmiToolbar.Text = "Панель інструментів";
            this.tsmiToolbar.Click += new System.EventHandler(this.tsmiToolbar_Click);
            //
            // tsmiStatusbar
            //
            this.tsmiStatusbar.Checked = true;
            this.tsmiStatusbar.CheckState = System.Windows.Forms.CheckState.Checked;
            this.tsmiStatusbar.Name = "tsmiStatusbar";
```

```
            this.tsmiStatusbar.Size = new System.Drawing.Size(200, 22);
            this.tsmiStatusbar.Text = "Рядок стану";
            this.tsmiStatusbar.Click += new System.EventHandler(this.tsmiStatusbar_Click);
            //
            // tssLine6
            //
            this.tssLine6.Name = "tssLine6";
            this.tssLine6.Size = new System.Drawing.Size(197, 6);
            //
            // tsmiBigIcon
            //
            this.tsmiBigIcon.Name = "tsmiBigIcon";
            this.tsmiBigIcon.Size = new System.Drawing.Size(200, 22);
            this.tsmiBigIcon.Text = "Великі значки";
            this.tsmiBigIcon.Click += new System.EventHandler(this.tsmiBigIcon_Click);
            //
            // tsmiSmallIcon
            //
            this.tsmiSmallIcon.Name = "tsmiSmallIcon";
            this.tsmiSmallIcon.Size = new System.Drawing.Size(200, 22);
            this.tsmiSmallIcon.Text = "Малі значки";
            this.tsmiSmallIcon.Click                    +=                    new
System.EventHandler(this.tsmiSmallIcon_Click);
            //
            // tsmiList
            //
            this.tsmiList.Name = "tsmiList";
            this.tsmiList.Size = new System.Drawing.Size(200, 22);
            this.tsmiList.Text = "Список";
            this.tsmiList.Click += new System.EventHandler(this.tsmiList_Click);
            //
            // tsmiDetailedInfo
            //
            this.tsmiDetailedInfo.Name = "tsmiDetailedInfo";
            this.tsmiDetailedInfo.Size = new System.Drawing.Size(200, 22);
```

```
            this.tsmiDetailedInfo.Text = "Більше інформації";
            this.tsmiDetailedInfo.Click += new
System.EventHandler(this.tsmiDetailedInfo_Click);
            //
            // tssLine7
            //
            this.tssLine7.Name = "tssLine7";
            this.tssLine7.Size = new System.Drawing.Size(197, 6);
            //
            // tsmiRefresh
            //
            this.tsmiRefresh.Name = "tsmiRefresh";
            this.tsmiRefresh.Size = new System.Drawing.Size(200, 22);
            this.tsmiRefresh.Text = "Оновити";
            this.tsmiRefresh.Click += new System.EventHandler(this.tsmiRefresh_Click);
            //
            // tsmiTool
            //
            this.tsmiTool.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.tsmiMonitor});
            this.tsmiTool.Name = "tsmiTool";
            this.tsmiTool.Size = new System.Drawing.Size(94, 21);
            this.tsmiTool.Text = "Інструменти";
            //
            // tsmiMonitor
            //
            this.tsmiMonitor.Name = "tsmiMonitor";
            this.tsmiMonitor.Size = new System.Drawing.Size(237, 22);
            this.tsmiMonitor.Text = "Моніторинг файлів/папок";
            this.tsmiMonitor.Click += new System.EventHandler(this.tsmiMonitor_Click);
            //
            // tsmiHelp
            //
```

```
            this.tsmiHelp.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.tsmiAbout});
            this.tsmiHelp.Name = "tsmiHelp";
            this.tsmiHelp.Size = new System.Drawing.Size(82, 21);
            this.tsmiHelp.Text = "Допомога";
            //
            // tsmiAbout
            //
            this.tsmiAbout.Name = "tsmiAbout";
            this.tsmiAbout.Size = new System.Drawing.Size(164, 22);
            this.tsmiAbout.Text = "Про програму";
            this.tsmiAbout.Click += new System.EventHandler(this.tsmiAbout_Click);
            //
            // tsMain
            //
            this.tsMain.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.tsbtnBack,
            this.tsbtnAdvance,
            this.tsbtnUpArrow,
            this.tssLine1,
            this.tscboAddress,
            this.tssLine2,
            this.tscboSearch});
            this.tsMain.Location = new System.Drawing.Point(0, 27);
            this.tsMain.Name = "tsMain";
            this.tsMain.Size = new System.Drawing.Size(984, 25);
            this.tsMain.TabIndex = 1;
            this.tsMain.Text = "toolStrip1";
            //
            // tsbtnBack
            //
            this.tsbtnBack.DisplayStyle                                    =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
```

```
            this.tsbtnBack.Image =
((System.Drawing.Image)(resources.GetObject("tsbtnBack.Image")));
            this.tsbtnBack.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.tsbtnBack.Name = "tsbtnBack";
            this.tsbtnBack.Size = new System.Drawing.Size(23, 22);
            this.tsbtnBack.Text = "toolStripButton1";
            this.tsbtnBack.Click += new System.EventHandler(this.tsbtnBack_Click);
            //
            // tsbtnAdvance
            //
            this.tsbtnAdvance.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.tsbtnAdvance.Image =
((System.Drawing.Image)(resources.GetObject("tsbtnAdvance.Image")));
            this.tsbtnAdvance.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.tsbtnAdvance.Name = "tsbtnAdvance";
            this.tsbtnAdvance.Size = new System.Drawing.Size(23, 22);
            this.tsbtnAdvance.Text = "toolStripButton1";
            this.tsbtnAdvance.Click += new System.EventHandler(this.tsbtnAdvance_Click);
            //
            // tsbtnUpArrow
            //
            this.tsbtnUpArrow.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.tsbtnUpArrow.Image =
((System.Drawing.Image)(resources.GetObject("tsbtnUpArrow.Image")));
            this.tsbtnUpArrow.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.tsbtnUpArrow.Name = "tsbtnUpArrow";
            this.tsbtnUpArrow.Size = new System.Drawing.Size(23, 22);
            this.tsbtnUpArrow.Text = "toolStripButton1";
            this.tsbtnUpArrow.Click += new
System.EventHandler(this.tsbtnUpArrow_Click);
            //
            // tssLine1
            //
```

```
            this.tssLine1.Name = "tssLine1";
            this.tssLine1.Size = new System.Drawing.Size(6, 25);
            //
            // tscboAddress
            //
            this.tscboAddress.AutoSize = false;
            this.tscboAddress.BackColor = System.Drawing.SystemColors.ControlLight;
            this.tscboAddress.Font = new System.Drawing.Font("Microsoft YaHei", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.tscboAddress.ForeColor = System.Drawing.Color.Black;
            this.tscboAddress.Name = "tscboAddress";
            this.tscboAddress.Overflow =
System.Windows.Forms.ToolStripItemOverflow.Never;
            this.tscboAddress.Size = new System.Drawing.Size(700, 25);
            this.tscboAddress.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.tscboAddress_KeyDown);
            //
            // tssLine2
            //
            this.tssLine2.Name = "tssLine2";
            this.tssLine2.Size = new System.Drawing.Size(6, 25);
            //
            // tscboSearch
            //
            this.tscboSearch.BackColor = System.Drawing.SystemColors.ControlLight;
            this.tscboSearch.Name = "tscboSearch";
            this.tscboSearch.Size = new System.Drawing.Size(170, 25);
            this.tscboSearch.Text = "Пошук";
            this.tscboSearch.Enter += new System.EventHandler(this.tscboSearch_Enter);
            this.tscboSearch.Leave += new System.EventHandler(this.tscboSearch_Leave);
            this.tscboSearch.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.tscboSearch_KeyDown);
            //
            // cmsMain
            //
```

```
            this.cmsMain.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.tsmiOpen,
            this.tsmiView1,
            this.tsmiRefresh1,
            this.tssLine3,
            this.tsmiCopy1,
            this.tsmiPaste1,
            this.tsmiCut1,
            this.tsmiDelete1,
            this.tsmiRename,
            this.tssLine4,
            this.tsmiNewFolder1,
            this.tsmiNewFile1,
            this.tssLine5,
            this.tsmiPrivilege1,
            this.tsmiProperties1});
            this.cmsMain.Name = "contextMenuStrip1";
            this.cmsMain.Size = new System.Drawing.Size(196, 286);
            this.cmsMain.Opening += new
System.ComponentModel.CancelEventHandler(this.cmsMain_Opening);
            //
            // tsmiOpen
            //
            this.tsmiOpen.Name = "tsmiOpen";
            this.tsmiOpen.Size = new System.Drawing.Size(195, 22);
            this.tsmiOpen.Text = "Відкрити";
            this.tsmiOpen.Click += new System.EventHandler(this.tsmiOpen_Click);
        }
```