

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 проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА СИСТЕМНОЇ УТИЛІТИ "ФАЙЛОВИЙ МЕНЕДЖЕР".
(назва теми)

DEVELOPMENT OF THE FILE MANAGER SYSTEM UTILITY

Виконав: студент(ка) 4 курсу, групи КНТ-119
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ШЕВЧЕНКО Я.О.

(ПРИЗВИЩЕ та ініціали)

Керівник СЕРДЮК С.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШЕВЧЕНКА Ярослава Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка системної утиліти «Файловий менеджер». Development of the File Manager System Utility.

керівник проєкту (роботи) к.т.н., доцент, СЕРДЮК Сергій Микитович,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові.)

затверджені наказом закладу вищої освіти від “04” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 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. Розробка структури програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Додатки.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СЕРДЮК С.М., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання 03 березня 2023 р.

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.	2 тиждень	Розділ 1
3	Розробка архітектури програми.	3 тиждень	Розділ 2
4	Розробка програми.	4–5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Розділ 5, Додатки
7	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Ярослав ШЕВЧЕНКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій СЕРДЮК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 107 с., 18 табл., 24 рис., 2 дод., 13 джерел.

ФАЙЛ, ФАЙЛОВА СИСТЕМА, ДИРЕКТОРІЯ, ОПЕРАЦІЇ З ФАЙЛАМИ, VISUAL STUDIO 2019, C#, ОПЕРАЦІЙНА СИСТЕМА, WINDOWS.

Об'єкт дослідження – файлова система операційної системи (ОС) Windows.

Предмет дослідження – методи взаємодії користувача з файловою системою.

Мета роботи – розробка програми, що забезпечує користувачу взаємодію з файловою системою ОС Windows.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування; мова програмування C#; середовище розробки Microsoft Visual Studio 2019; персональний комп'ютер з процесором Intel Core 7 Duo під управлінням ОС Microsoft Windows 10.

Результати. Розроблено програмне забезпечення, яке надає користувачу зручний інтерфейс для організації роботи з файловою системою та файлами: створення, видалення, копіювання, вставки, вирізання, перейменування та відкриття файлів.

Галузь використання – комп'ютерні системи з ОС Windows.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 107 pages, 24 figures, 18 tables, 2 appendixes, 13 sources.

FILE, FILE SYSTEM, DIRECTORY, FILE OPERATIONS, VISUAL STUDIO 2019, C#, OPERATING SYSTEM, WINDOWS.

The object of study is a file system of the Windows operating system (OS).

The subject of study are methods of user interaction with the file system.

The purpose of the work is a development of a program that provides user interaction with the Windows OS file system.

Materials, methods and technical means: object-oriented programming; programming languages C#; .NET software platform; Microsoft Visual Studio 2019 development environment; a personal computer with an Intel Core 7 Duo processor running Microsoft Windows 10.

Results. Software has been developed, which provides the user with a convenient interface for organizing work with the file system and files: creating, deleting, copying, inserting, cutting, renaming, and opening files.

The field of use are computer systems with Windows OS.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Аналіз програмних аналогів.....	10
1.1.1 Програма «Total Commander».....	10
1.1.2 Програма «Far Manager»	10
1.1.3 Провідник Windows (Windows Explorer)	10
1.2 Розробка технічного завдання.....	16
1.2.1 Підстава для розробки.....	16
1.2.2 Мета розробки та її призначення	16
1.2.3 Основні вимоги до програмного продукту	16
1.2.4 Вимоги до технічних засобів, операційної системи і мови програмування	18
1.2.5 Вимоги до інформаційної та програмної сумісності	19
1.2.6 Вимоги до програмної документації	19
1.2.7 Порядок приймання і тестування.....	19
2 Розробка структури програмного забезпечення.....	20
2.1 Аналіз технічного завдання.....	20
2.2 Розробка структури проєкту.....	20
2.3 Розробка функціональної схеми застосунку.....	22
3 Основні рішення щодо реалізації компонентів системи	24
3.1 Огляд особливостей мови програмування	24
3.2 Огляд особливостей середовища розробки	26
3.3 Огляд використаних бібліотек	29
3.4 Основні рішення щодо розроблення проєкту застосунку	31
3.4.1 Клас MainForm.....	33
3.4.2 Клас AboutForm	38
3.4.3 Клас DiskInfoForm	39

3.4.4 Клас ErrorForm	40
3.4.5 Клас FilePropertiesForm	41
3.4.6 Клас RenamingForm	42
3.4.7 Клас HelpForm.....	43
3.4.8 Клас FileObject	44
3.4.9 Клас Style	45
3.4.10 Клас MenuStripRendererStandart	47
3.4.11 Клас ColsStandart.....	48
3.4.12 Клас MenuStripRendererBlack	49
3.4.13 Клас ColsBlack.....	49
3.4.14 Клас MenuStripRendererLight.....	50
3.4.15 Клас ColsLight	51
3.5 Клас Основні рішення щодо розробки інтерфейсу	52
4 Керівництво програміста	58
4.1 Призначення та умови застосування програми	58
4.2 Характеристики програми	58
4.3 Структура програми	58
4.4 Вхідні та вихідні дані	59
4.4.1 Вхідні дані	59
4.4.2 Вихідні дані	60
4.5 Повідомлення програмісту	60
5 Керівництво оператора	62
5.1 Призначення програми	62
5.2 Умови виконання програми.....	62
5.3 Виконання програми	62
5.4 Повідомлення оператору	67
Висновки	69
Перелік джерел посилання	70
Додаток А. Текст програми	72
Додаток Б. Слайди презентації.....	99

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interfaces;
IDE	– Integrated development environment;
VS	– Visual Studio;
Гб	– гігабайт;
ГГц	– гігагерц;
Мб	– мегабайт;
ООП	– об’єктно-орієнтоване програмування;
ОС	– операційна система;
ПЗ	– програмне забезпечення;
ПК	– персональний комп’ютер.
ТЗ	– технічне завдання.

ВСТУП

Сучасні технології комп'ютерів є необхідними у сьогоденні. За їх допомогою впроваджують інновації та покращують інформаційні системи, досліджуючи всі сфери життя людини. Майже кожен день ми працюємо за персональним комп'ютером (ПК), а також з різними файлами, такими як фотографії, таблиці Excel, музика та інше.

Через великий обсяг роботи з файлами, були створені файлові менеджери (ФМ). ФМ є програмами, які спрощують та прискорюють роботу з файлами, дозволяючи створювати, копіювати, перейменовувати, переміщувати та видаляти файли.

Існує дуже багато різних ФМ. Як правило сучасні ОС мають власний вбудований ФМ. Більшість операцій з файлами може бути виконана вручну, але це вимагає багатого часу та наявності поглиблених знань у роботі з комп'ютером.

Створення ФМ для ОС Windows є актуальною темою, яка дозволяє краще організувати роботу з файловою системою.

Метою дипломної кваліфікаційної роботи бакалавра є створення ФМ зі зрозумілим та зручним інтерфейсом для покращення роботи з файловою системою ОС Windows 10.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз програмних аналогів

Предметом дослідження даної роботи є методи взаємодії користувача з файловою системою ОС Windows. Основна проблема при роботі користувача з ПК полягає в роботі з великою кількістю файлів, що займає велику кількість часу, якщо виконувати все в ручну. Дана проблема особливо актуальна для звичайних користувачів, які не мають поглиблених знань у використанні комп'ютера. Крім того, більшість сучасних видів діяльності пов'язані з використанням комп'ютера десь у дорозі. тоді як інші повністю залежать від нього. Робота з великою кількістю файлів може бути неприємною, особливо коли час відіграє важливу роль. На щастя, за допомогою спеціалізованих програм, таких як ФМ, керування файлами стає простою задачею.

Метою роботи є розроблення програми ФМ, який повинен надавати можливість перегляду змісту файлової системи, проведення стандартних операцій: створення, видалення, копіювання, вставки, вирізання, перейменування, відкриття файлів, а також зручне налаштовування інтерфейсу програми.

Отримавши завдання до дипломного проєкту щодо існуючих ФМ для ОС Windows, був досліджений та опрацьований ринок представлених сервісів та програм в даній предметній області. Було виявлено що існує велика кількість аналогів, і кожен з них має свої функції, недоліки та переваги.

Проаналізувавши вже існуючі аналоги, було обрано найбільш розповсюджені, схожі за функціоналом та змістом програми: Total Commander, Far Manager та Провідник Windows (Windows Explorer). Розглянемо їх більш детально.

1.1.1 Програма «Total Commander»

Total Commander – популярний двопанельний файловий менеджер для операційних систем Windows, Windows CE, Windows Mobile і Android [1]. Двопанельний інтерфейс цього ФМ набув масового поширення завдяки зручності, компактності та доволі високому рівню функціональних можливостей.

Основні функції ФМ [2]:

- великий набір операцій з файлами та папками;
- робота з архівами, що обробляються як підкаталоги. Користувач може легко скопіювати файли до та з архівів. Програма підтримує безліч архівів (zip, rar, tar, gz і т.д.) і всі операції з ними. Вбудований в PKZIP пакувальник, підтримує довгі імена файлів;
- розрізання та збирання файлів великого об'єму;
- групове перейменування файлів та синхронізація каталогів;
- вбудований переглядач файлів для перегляду файлів будь-якого розміру у шістнадцятковому, двійковому чи текстовому форматі;
- розширений пошук файлів, включаючи пошук тексту у будь-яких файлах, у тому числі на кількох дисках;
- можливість підключати плагіни;
- клавіатурні комбінації для перегляду, редагування, копіювання та видалення файлів (можливість повного керування без використання миші).

Скріншот прикладу роботи програми Total Commander наведено на рисунку 1.1.

До переваг ФМ «Total Commander» слід віднести:

- інформативний інтерфейс;
- підтримка української мови;
- підтримка плагінів;
- вбудований переглядач файлів;
- можливість повного керування без використання миші;
- двопанельний інтерфейс, що забезпечує можливість роботи одразу з двома папками в одному вікні.

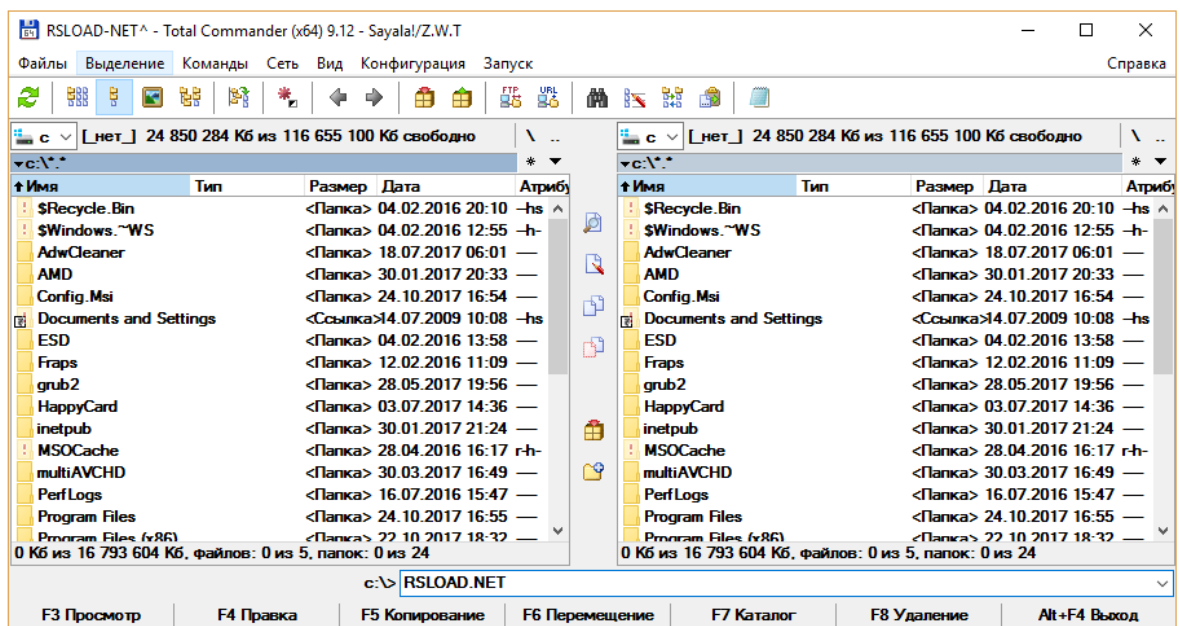


Рисунок 1.1 – Скріншот прикладу роботи «Total Commander»

Недоліками ФМ «Total Commander» є:

- незручний для недосвідченого користувача дизайн;
- він платний.

1.1.2 Програма «Far Manager»

Far Manager – консольний файловий менеджер для операційних систем сімейства Microsoft Windows та Linux [3].

Основні функції програми Far Manager:

- операції з файлами та папками;
- керування без використання миші;
- вибір режимів відображення;
- двопанельний інтерфейс;
- сортування елементів;
- робота з архівами (в стандартній поставці – тільки перегляд);
- фільтрація елементів;

- запис макросів;
- можливість підключати плагіни;
- перекодування у вбудованому редакторі.

Скріншот прикладу роботи програми Far Manager наведено на рисунку

1.2.

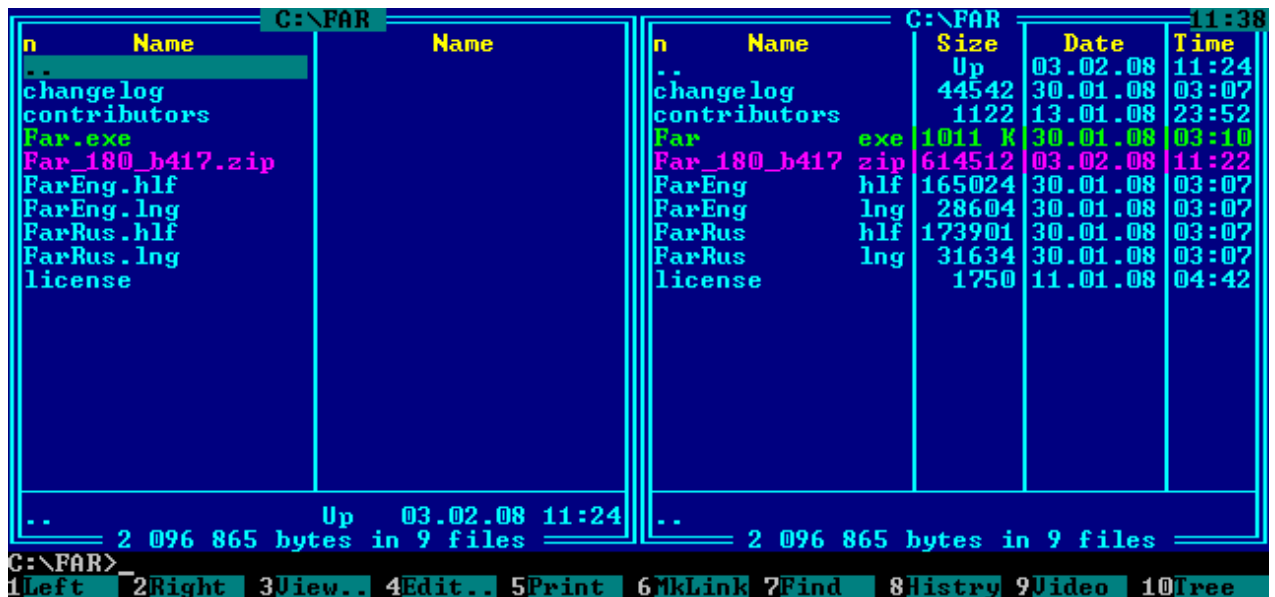


Рисунок 1.2 – Скріншот прикладу роботи програми «Far Manager»

Переваги Far Manager:

- двопанельний інтерфейс – можливість роботи одразу з двома папками в одному вікні;
- підтримка плагінів;
- підтримка великої кількості кодових таблиць;
- багатомовність та легка локалізація;
- цілком безкоштовний.

Недоліки Far Manager:

- складність у налаштуванні інтерфейсу;
- незручний для недосвідченого користувача інтерфейс.

1.1.3 Провідник Windows (Windows Explorer)

Провідник Windows – навігаційний файловий менеджер, що міститься у каталозі операційної системи Windows [4]. Це вбудований застосунок, який реалізує графічний інтерфейс доступу користувача до файлів операційної системи Microsoft Windows.

Основні функції ФМ:

- операції з файлами та папками;
- швидкий доступ до часто відвідуваних місць;
- перемикання виду папки;
- запуск командного рядка з повними правами у потрібній папці;
- гарячі клавіші.

Скріншот прикладу роботи програми наведено на рис. 1.3.

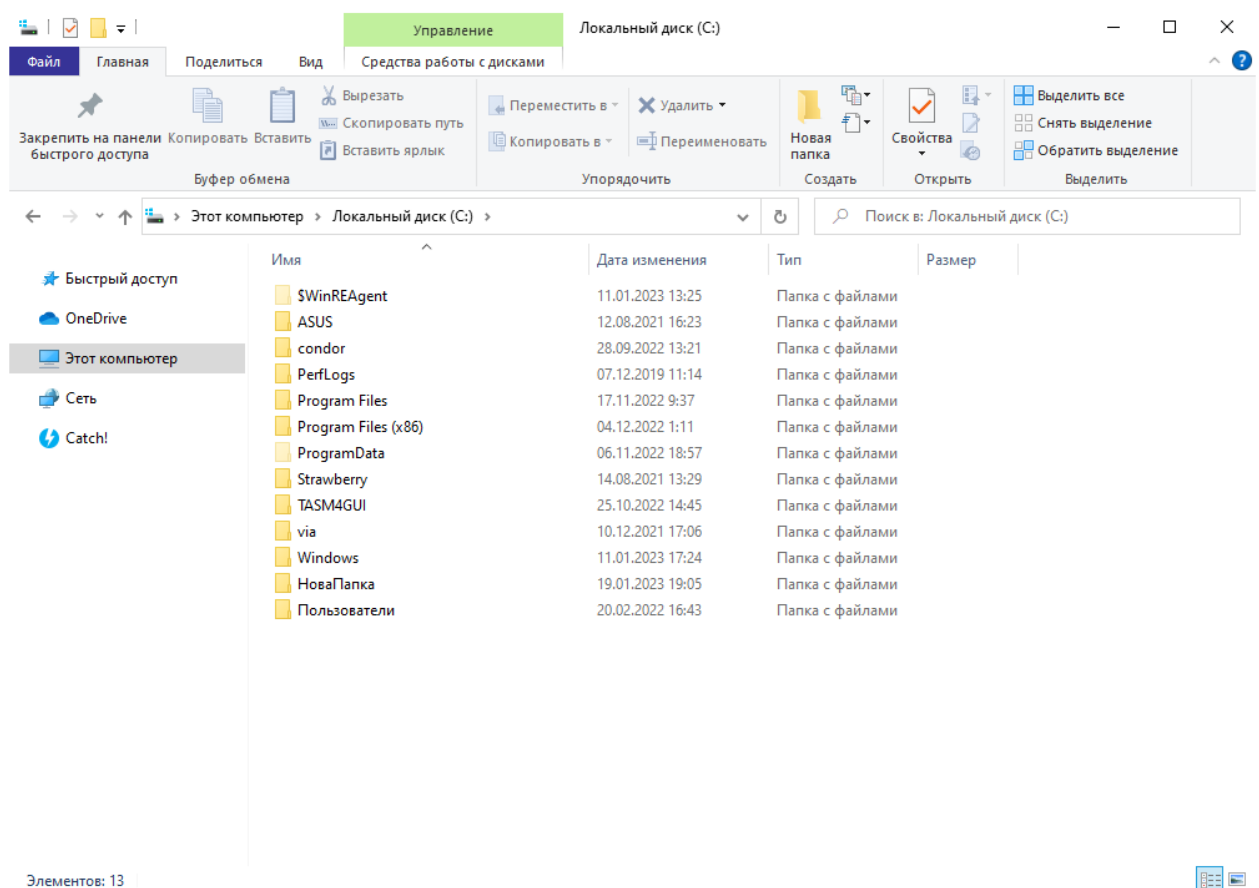


Рисунок 1.3 – Скріншот прикладу роботи програми «Провідник Windows»

Переваги ФМ «Провідник Windows»:

- дружній для початківця інтерфейс.

Недоліки:

- відсутність двопанельного режиму;
- невеликий функціонал;
- не підтримує плагіни.

Для порівняння всіх розглянутих аналогів було створено таблицю 1.1. В цій таблиці було порівняно: Total Commander, Far Manage та Провідник Windows за найбільш важливими характеристиками з точки зору забезпечення взаємодії користувача з файловою системою ОС Windows.

Таблиця 1.1 – Порівняння аналогів

Характеристика	Total Commander	Far Manager	Провідник Windows
Можливість проводити операції з файлами	+	+	+
Підтримка плагінів	+	+	-
Безкоштовність	-	+	+
Дружній дизайн інтерфейсу	-	-	+
Можливість зручно налаштувати інтерфейс	+	-	-
Двопанельний інтерфейс	+	+	-

Згідно з результатами порівняння в таблиці 1.1 можна побачити, що Total Commander та Far Manager мають обширний функціонал, на відміну від Провідника Windows. Але також треба зазначити, що програма Far Manager має не зовсім зрозумілий інтерфейс для звичайного користувача на відміну від Провідника Windows. Total Commander є більш зрозумілим ніж Far Manager для користувачів, але так як ця програма має двопанельний інтерфейс, то для багатьох користувачів, які не звикли до такого виду інтерфейсів, потрібно

буде додаткове звикання до нього.

Таким чином прототипом було обрано – Провідник Windows, як є найбільш дружнім ФМ для користувача. В результаті проведеного аналізу було отримано інформацію, необхідну для розробки технічного завдання на програму, що розроблюється.

1.2 Розробка технічного завдання

1.2.1 Підстава для розробки

Підставою для розробки служить завдання на дипломну кваліфікаційну роботу бакалавра на тему "Розробка системної утиліти «Файловий менеджер»", затверджене наказом № 87 від 04 квітня 2023 р. по НУ «Запорізька політехніка».

1.2.2 Мета розробки та її призначення

Метою є розробка програмного застосунку, що призначений для забезпечення взаємодії користувача з файловою системою ОС Windows.

1.2.3 Основні вимоги до програмного продукту

Програмне забезпечення повинно мати один тип користувачів без розподілення на інші групи. Вони можуть:

- переглядати зміст файлової системи;
- проводити стандартні операції: створення, видалення, копіювання, вставки, вирізання, перейменування, відкриття файлів;
- налаштовувати інтерфейс програми.

Програмне забезпечення, що розроблюється має надавати користувачеві наступні можливості:

- перегляд змісту файлової системи, тобто перегляд носіїв даних підключених до ПК та їх змісту;

- створення файлу або папки;
- відкриття файлу або папки;
- копіювання файлу або папки;
- вставки файлу або папки;
- вирізання файлу або папки;
- перейменування файлу або папки;
- налаштовування інтерфейсу програми (зміна кольорової палітри інтерфейсу, мови);
- відображення шляху до поточної директорії;
- відображення томів, їх обсягу вільного та занятого простору;
- відображення інформації про файл (назва, тип файлу, об'єм занятого простору, дата та час останньої зміни).

Вхідними даними в програмі є вміст файлової системи та дані, яку вводять користувачі. Вміст файлової системи буде зберігатися у списку структур. Структура буде складатися з тих елементів, як:

- назва файлу;
- тип файлу;
- розмір файлу;
- дата створення файлу.

Вихідними даними в програмі буде інформація, що виведена на екран користувачу.

Основною вимогою до вихідних даних є зрозумілість виведеної інформації для користувача та її змістовність.

До тимчасових характеристик відноситься:

- список структур, що створений для отримання вмісту файлової системи;
- файл, в якому буде зберігатись список вмісту поточної директорії;
- стек, в якому буде записуватися шлях до поточної директорії.

Використання цих засобів дозволить в процесі роботи з програмою оновлювати дані, які повинні відображатися в інтерфейсі.

1.2.4 Вимоги до технічних засобів, операційної системи і мови програмування

Програма, що розробляється у вигляді Desktop-застосунку має бути написана на мові C# у середовищі розробки Microsoft Visual Studio 2019. Програма призначена для роботи на персональному комп'ютері під керуванням ОС Windows 10.

Для забезпечення нормальної роботи ОС Windows 10 згідно [13] мають бути виконані наступні системні вимоги:

- процесор із тактовою частотою 1 ГГц або швидший чи SoC;
- 1 гігабайт оперативної пам'яті (для 32-розрядної версії) або 2 ГБ (для 64-розрядної версії);
- обсяг вільного дискового простору 16 ГБ (для 32-розрядної версії) або 20 ГБ (для 64-розрядної версії);
- графічна плата із підтримкою DirectX 9 або новішої версії з драйвером WDDM 1.0 або новішим;
- дисплей: 800 x 600.

Також, щоб забезпечити функціонування програми ФМ необхідні наступні засоби:

- 10 Мб вільного місця на жорсткому диску для встановлення програми;
- маніпулятори вводу (клавіатура, миша).

1.2.5 Вимоги до інформаційної та програмної сумісності

Інформаційні структури повинні бути розроблені згідно з визначеними структурами вхідних даних. Отримані вхідні коди повинні бути написані мовою C#. Середовище розробки програмного продукту Microsoft Visual Studio.

1.2.6 Вимоги до програмної документації

Програмна документація має включати: керівництво програміста, керівництво оператора, опис програми та тексти основних програмних модулів.

1.2.7 Порядок приймання і тестування

Тестування даного застосунку проводиться у вигляді виконання усіх можливих сценаріїв роботи оператора з файловою системою.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз технічного завдання

Згідно розробленого у п.1.2.3 технічного завдання (ТЗ) було виявлено основні функції, які повинно виконувати програмне забезпечення, а саме:

- перегляд змісту файлової системи, тобто перегляд носіїв даних підключених до ПК та їх змісту;
- створення файлу або папки;
- відкриття файлу або папки;
- копіювання файлу або папки;
- вставки файлу або папки;
- вирізання файлу або папки;
- перейменування файлу або папки;
- налаштовування інтерфейсу програми (зміна кольорової палітри інтерфейсу, мови)
- відображення шляху до поточної директорії;
- відображення дисків, їх обсягу вільного та занятого простору
- відображення інформації про файл (назва, тип файлу, об'єм занятого простору, дата та час останньої зміни).

Також було визначено вхідні та вихідні дані.

Вхідними даними в програмі є вміст файлової системи та дані, що вводять користувачі.

Вихідні даними в програмі – це інформація яка виведена на екран компютеру за допомогою людино-машинного інтерфейсу.

2.2 Розробка структури проєкту

Структурною називають схему, що відбиває склад і взаємозв'язки по управлінню частинами програмного продукту, який розробляється.

«Підсистема для обробки виняткових ситуацій» – забезпечує обробку помилок та коректне завершення роботи, використовуючи обробку виняткових ситуацій.

«Підсистема для отримання інформації про програму» – забезпечує вивід інформації про версію та розробника програми ФМ.

«Підсистема для отримання довідки» – забезпечує вивід інформації про призначення та функціональність програми ФМ.

«Компонент для налаштування стилю інтерфейсу» – забезпечує можливість змінити стиль інтерфейсу згідно вподобань користувача.

2.3 Розробка функціональної схеми застосунку

Функціональна схема – це схема взаємодії компонентів програмного забезпечення з описом інформаційних потоків, складу даних в потоках і вказівкою використовуваних файлів і пристроїв. Функціональну схему програмного забезпечення, що розроблюється наведено на рис. 2.2.

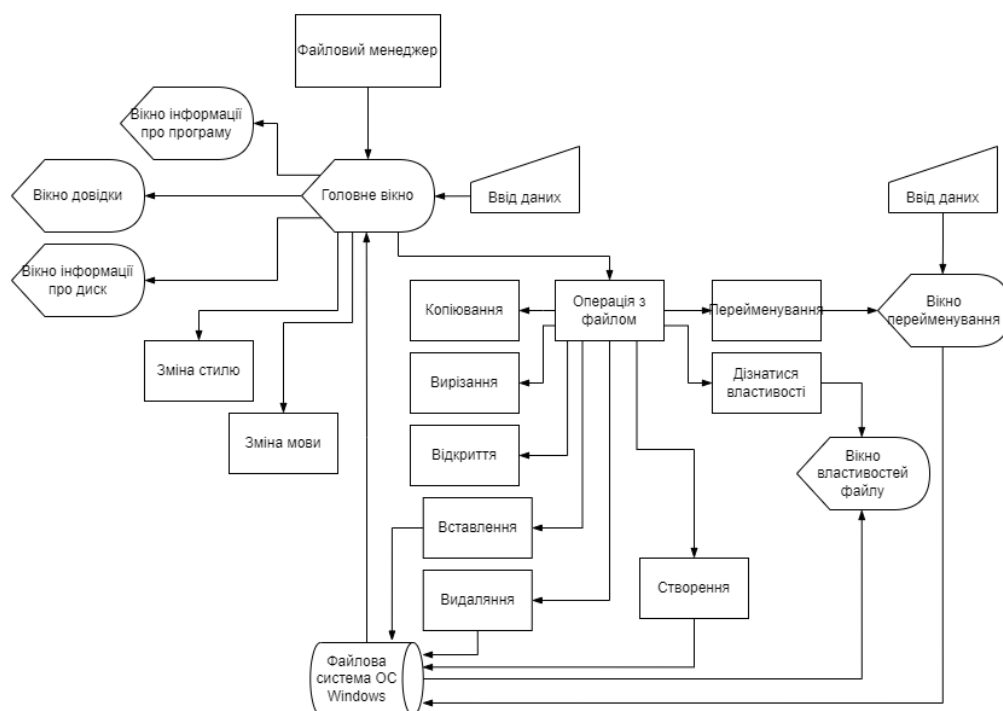


Рисунок 2.2 – Функціональна схема застосунку

Таким чином у даному розділі пояснювальної записки було проаналізовано технічне завдання, розроблено структурну та функціональну схеми застосунку.

Отже, було розроблено концептуальну модель для програмного забезпечення ФМ «FreeFile».

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

В даному розділі розглянуті основні особливості програмних засобів, за допомогою яких реалізовано дипломну кваліфікаційну роботу. В процесі виконання даного програмного проєкту було обрано середовище розробки, мову програмування та розроблена ієрархія класів. Було визначено основні переваги обраних програмних засобів, які найбільше підходять для виконання поставлених задач та цілей програмного рішення з назвою «FreeFile».

Для вибору вищенаведеного були враховані наступні критерії: зручність, функціональність, доступність, зрозумілість, легкість у використанні.

3.1 Огляд особливостей мови програмування

Мова програмування – це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам. Мови програмування використовуються для створення програм, які контролюють поведінку машин, та для запису алгоритмів. Інакше кажучи, мова програмування – це система позначень для опису алгоритмів і структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми та дії, які виконує виконавець (комп'ютер) під її управлінням [5].

В якості мови програмування було обрано C#. Тому що саме ця мова ідеально підходить для розробки десктопного програмного забезпечення та надає багато можливостей для розробки дружнього та зручного інтерфейсу.

Також C# застосовується для розробки десктопних Windows-застосунків. Наприклад, Microsoft Visual Studio, Paint.NET написані на C#.

Платформа.NET, точніше, Windows Forms лежить у основі Skype, Microsoft Office, Photoshop [6].

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Вона розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research [7].

Об'єктно-орієнтоване програмування (ООП) – це одна з парадигм програмування, яка має три загальні властивості:

- інкапсуляція – це властивість, яка дозволяє об'єднати дані та методи в класі, та приховати деталі реалізації від користувача;
- спадкування – це властивість, яка дозволяє описати новий клас на основі вже існуючого класу;
- поліморфізм – це властивість, яка дозволяє одному інтерфейсу отримувати доступ до спільного класу дій.

Об'єктно-орієнтована мова програмування – мова, що побудована на принципах об'єктно-орієнтованого програмування, яка об'єднує в собі методи та поля.

C# є дуже близьким родичом мови програмування Java. У 1995 році компанія Sun Microsystems створила мову Java. Її розробили з урахуванням граматики C++. Головною «фішкою», що відрізняє цю мову від інших, стало ось що. До перетворення на машинний код програма на Java спочатку перекладалася на байт-код. Це робило мову Java незалежною від апаратного забезпечення машини. А ще розробники додали туди підтримку вебу, щоб можна було взаємодіяти з інтернетом: програмувати клієнт-серверні програми, отримувати дані з інтернету та обмінюватися ними.

Microsoft треба було скласти конкуренцію Sun і також працювати з вебом. Мільйони рядків коду в Microsoft були написані внутрішніми мовами (в основному C і C++). Тому просто так перейти на Java було неможливо.

Microsoft вирішили, користуючись своїм панівним становищем на ринку, створити свій власний аналог Java — мову, для якої корпорація буде повноцінним власником. Ця новостворена мова отримала назву C#. Вона

успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду (MSIL) і більшої безпеки вихідного коду програм, а також врахувала досвід використання програм на Java.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проєктів [6].

Переваги C# наступні [7]-[8]:

- одним із найважливіших плюсів є наявність великої кількості бібліотек та шаблонів, що дозволяють не витрачати час та винахід свого власного шаблону чи іншого;
- ще до плюсів можна віднести строгу типізацію, яка дозволяє захиститися від помилок;
- C# відрізняється надійністю та елегантністю. Елегантністю C# відрізняється через велику різноманітність синтаксичних конструкцій;
- мова входить в десятку лідируючих напрямків, затребуваних на ринку;
- мова створена так, щоб дати розробникові максимальний контроль над усіма аспектами виконання програми;
- мова активно розвивається, нові поліпшення з'являються набагато швидше, ніж в інших мовах програмування;
- об'єктно-орієнтована, проста і в той же час потужна мова програмування, яка дозволяє розробникам створювати багатофункціональні програми.

Саме через велику кількість переваг та популярність у використанні для розробки ФМ було обрано мову програмування C#.

3.2 Огляд особливостей середовища розробки

Інтегроване середовище розробки – комплексне програмне рішення для розробки програмного забезпечення [9].

Середовищем розробки програмного продукту було обрано Microsoft Visual Studio. Саме це середовище надає багато функцій і можливостей для роботи з C#.

Visual Studio – інтегроване середовище розробки програмного забезпечення від фірми Microsoft. Дане середовище дозволяє створювати різноманітні програмні продукти: консольні програми, програми з графічним інтерфейсом, наприклад віконні додатки Windows Forms, а також Web-застосунки тощо.

.NET Framework – це програмна основа, яка в основному працює в Microsoft Windows. Він містить велику бібліотеку та підтримує кілька мов програмування, що забезпечує взаємодію мов (кожна мова може використовувати код, написаний іншими мовами). Бібліотека .NET доступна для всіх мов програмування, які підтримує .NET.

.NET Framework – це революційна платформа, створена Microsoft для розробки програм. .NET Framework розроблено таким чином, що його можна використовувати будь-якою мовою, включаючи C#, а також C++, Visual Basic, JScript і навіть старіші мови [10].

.NET і Visual Studio дозволяють розробляти сучасні бізнес-застосунки для Windows, орієнтовані на обробку даних за допомогою платформи Windows Forms, WPF та інших.

Windows Forms – інтерфейс програмування програм (API), який відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Цей інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows.

Windows Forms – це платформа інтерфейсу користувача для створення класичних програм Windows. Вона забезпечує один з найефективніших способів створення класичних програм за допомогою візуального конструктора в Visual Studio. Такі функції, як розміщення візуальних елементів керування шляхом перетягування, полегшують створення класичних застосунків [11].

У Windows Forms можна розробляти графічно складні програми, які просто розгортати, оновлювати і з якими зручно працювати як в автономному режимі, так і в мережі. — це платформа інтерфейсу користувача для створення класичних програм Windows. Вона забезпечує один з найефективніших способів створення класичних програм за допомогою візуального конструктора в Visual Studio. Такі функції, як розміщення візуальних елементів керування шляхом перетягування, полегшують створення класичних застосунків [11]. У Windows Forms можна розробляти графічно складні програми, які просто розгортати, оновлювати і з якими зручно працювати як в автономному режимі, так і в мережі.

Усередині .NET Framework Windows Forms реалізується у межах простору імен System.Windows.Forms.

Microsoft Visual Studio має багато популярних функції, які облегшують роботу в цьому середовищі, наприклад [12]:

- хвилясті лінії та швидкі дії. Хвилясті лінії позначають помилки або потенційні проблеми коду під час введення. Ці візуальні підказки допомагають негайно усунути проблеми, не чекаючи появи помилок під час складання чи виконання. якщо навести вказівник на хвилясту лінію, на екран буде виведено додаткові відомості про помилку. Також у полі зліва може відображатися лампочка, що вказує на наявність відомостей про швидкі дії для усунення помилки;

- очищення коду. Можна одним натисканням кнопки відформатувати код та застосувати до нього виправлення, запропоновані параметрами стилю коду. Очищення коду зараз доступне лише для коду с# допомагає усувати проблеми в коді перед переходом до його перевірки;

- рефакторинг включає такі операції, як інтелектуальне перейменування змінних, вилучення одного або декількох рядків коду в новий метод і зміна порядку параметрів методів;

- intellisense – це набір можливостей, що відображають відомості про код безпосередньо в редакторі та в деяких випадках автоматично створюють

невеликі уривки коду. Фактично, це вбудована у редактор базова документація, яка позбавляє необхідності шукати інформацію в інших джерелах;

– codeLens допомагає знаходити посилання на код, зміни коду, пов'язані з кодом помилки, робочі елементи, перевірки коду та модульні тести – не виходячи з редактора.

3.3 Огляд використаних бібліотек

Бібліотеки, що було використано при створенні програмного забезпечення описані в табл. 3.1

Таблиця 3.1 – Використані бібліотеки

Назва	Опис	Використання
1	2	3
System	Містить фундаментальні та базові класи, що визначають типи значень і посилань, що часто використовуються, події та обробники подій, інтерфейси, атрибути та винятки обробки.	Дозволяє використовувати базові елементи.
Collections.Generic	Містить інтерфейси та класи, що визначають універсальні колекції, які дозволяють користувачам створювати строго типізовані колекції, що забезпечують підвищену продуктивність та безпеку типів у порівнянні з неуніверсальними строго типізованими колекціями.	Дозволяє використовувати такий клас, як List<T>.

Продовження табл. 3.1.

1	2	3
ComponentModel	Надає класи, що використовуються для реалізації поведінки компонентів та елементів керування під час розробки та виконання. Цей простір імен містить базові класи та інтерфейси для реалізації атрибутів та перетворювачів типів, прив'язки до джерел даних та ліцензування компонентів.	Дозволяє взаємодіяти з інтерфейсом
Diagnostics	Надає класи, що дозволяють здійснювати взаємодію з системними процесами, журналами подій та лічильниками продуктивності.	Дозволяє викликати консоль
Drawing	Надає доступ до основних графічних функцій	Ручне малювання елементів форми
IO	Надає статичні методи створення, копіювання, видалення, переміщення та відкриття одного файлу, а також допомагає при створенні об'єктів FileStream.	Для зчитування та запису у файли
Text	Містить класи, які представляють кодування ASCII та Юнікоду; абстрактні базові класи для перетворення блоків знаків на блоки байтів і назад	Дозволяє змінювати кодування при зчитуванні файлів

Продовження табл. 3.1.

1	2	3
Windows.Forms	Містить класи для створення програм Windows, які дозволяють найбільш ефективно використовувати розширені можливості інтерфейсу користувача, доступні в операційній системі Microsoft Windows.	Дозволяє реалізовувати інтерфейс користувача
Cells	Aspose.Cells для .NET підтримує популярні формати файлів електронних таблиць (XLS, XLSX, XLSM, XLSB, XLTX, XLTM, CSV, SpreadsheetML, ODS).	Створення файлів розширення .xlsx
System.Windows.Forms.DataVisualization.Charting	Містить методи та властивості для елемента керування формами Chart Windows.	Дозволяє створювати стилі для діаграми
Guna.U12 WinForms	Пакет для створення інтерфейсу користувача	Додає більше можливостей для змінювання дизайн елементів форми

3.4 Основні рішення щодо розроблення проєкту застосунку

В процесі розробки було створено наступні класи:

- MainForm – реалізує головну форму та основний функціонал ПЗ;
- AboutForm – реалізує форму з інформацією про ПЗ;
- DiskInfoForm – реалізує форму з інформацією про диск;

- `ErrorForm` – реалізує форму для обробки виняткових ситуацій;
- `FilePropertiesForm` – реалізує форму з інформацією про файл або папку;

- `RenamingForm` – реалізує форму для перейменування файлу;
- `HelpForm` – реалізує форму з довідкою;
- `FileObject` – реалізує об'єкт, який зберігає інформацію про файл;
- `Style` – реалізує основні рішення щодо налаштування теми форм

ПЗ;

- `MenuStripRendererStandart` – реалізує відмальовування елементу `MenuStrip` у стандартній темі;

- `ColsStandart` – задає налаштування стандартної теми для елементу `MenuStrip`;

- `MenuStripRendererBlack` – реалізує відмальовування елементу `MenuStrip` у темній темі;

- `ColsBlack` – задає налаштування темної теми для елементу `MenuStrip`;

- `MenuStripRendererLight` – реалізує відмальовування елементу `MenuStrip` у світлій темі;

- `ColsLight` – задає налаштування світлої теми для елементу `MenuStrip`.

В процесі розробки було створено наступні структури:

- `DiskObject` – реалізує збереження інформації про диск;
- `ClipboardObject` – реалізує збереження інформації буфера обміну;
- `DataAndTime` – реалізує збереження інформації про дату та час.

Для моделювання структури системи та класової ієрархії було розроблено діаграму класів, яка представлена на рис. 3.1

Розглянемо призначення основних класів програми більш детально.

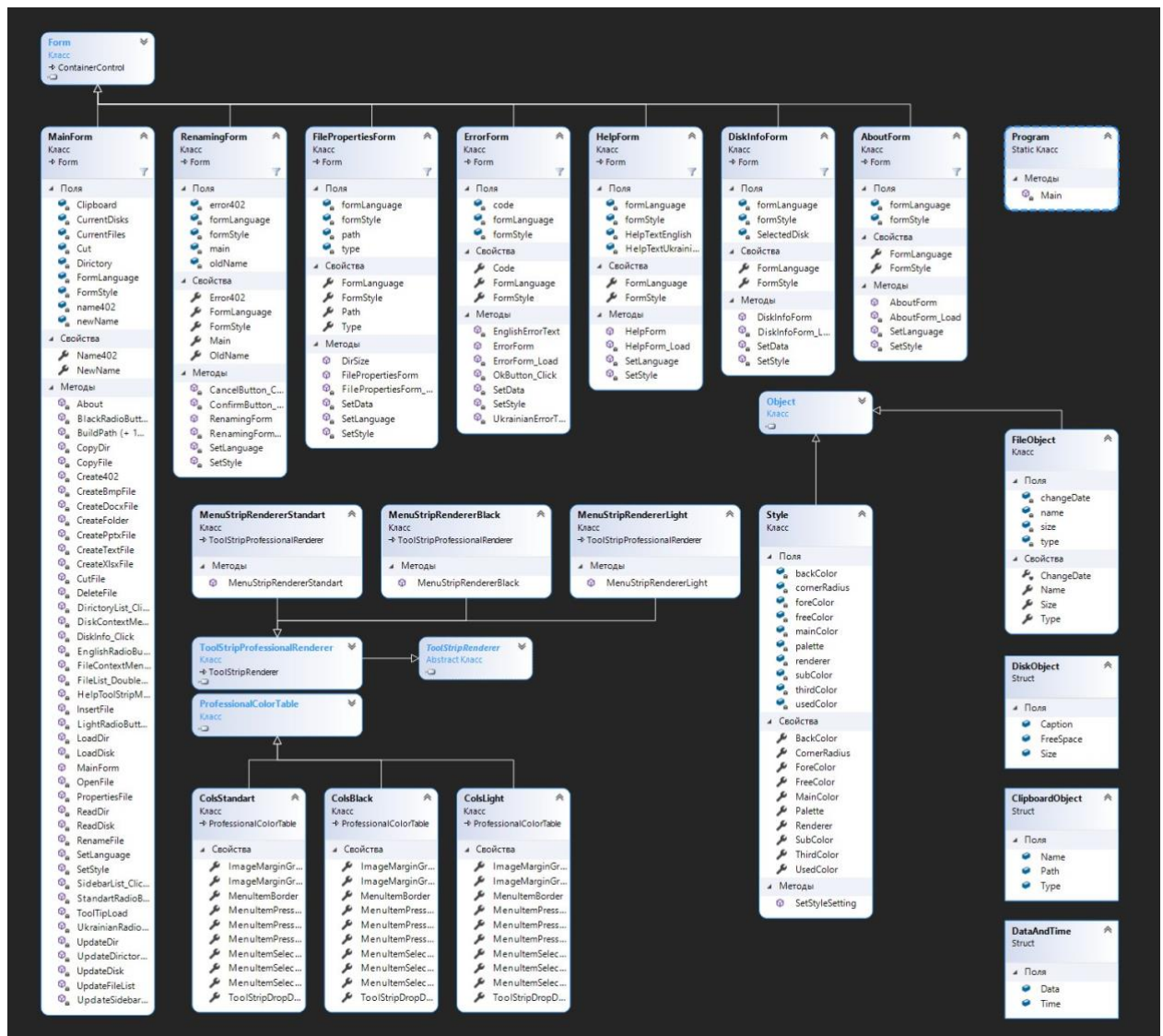


Рисунок 3.1 – Діаграма класів

3.4.1 Клас MainForm

Ціль класу MainForm полягає у реалізації основного функціоналу застосунку, а саме:

- відображення головного вікна;
- операції з файлами;
- технічні функції (Робота з дирикторією та дисками).

Дані та методи класу наведені у таблиці 3.2.

Таблиця 3.2 – Опис полів та методів класу MainForm

Поля та методи класу	Опис
1	2
<i>private:</i>	
List<string> Dirictory	Шлях до поточної директорії
List<FileObject> CurrentFiles	Зміст поточної директорії
List<DiskObject> CurrentDisks	Підключені диски
List<ClipboardObject> Clipboard	Буфер обміну
bool Cut	Прапорець для вирізання
string newName	Змінна для перейменування
string name402	Змінна для перейменування при однакових іменах
string FormStyle	Поточна тема форми
string FormLanguage	Поточна мова форми
bool LoadDir(List<string> Dir)	Завантаження до файлу змісту поточної директорії
bool ReadDir(string Path)	Зчитування файлу зі змістом поточної директорії
bool UpdateDir()	Оновлення змісту поточної директорії
bool LoadDisk()	Завантаження до файлу списку підключених дисків
bool ReadDisk(string Path)	Зчитування файлу зі списком підключених дисків
bool UpdateDisk()	Оновлення підключених дисків
bool UpdateFileList(List<FileObject> Files)	Оновлення ListView зі змістом поточної директорії
bool UpdateDirictoryList(List<string> Dir)	Оновлення ListView зі шляхом до поточної директорії

Продовження табл. 3.2.

1	2
bool UpdateSidebarList(List<DiskObject> Disks)	Оновлення ListView з підключеними дисками
private void FileList_DoubleClick(object sender, EventArgs e)	Реалізація реакції на подвійний клік на папку або файл
private void DirectoryList_Click(object sender, EventArgs e)	Реалізація реакції на натискання на елемент ListView зі шляхом до поточної директорії
private void SidebarList_Click(object sender, MouseEventArgs e)	Відкриття диска з меню
private void SidebarList_Click(object sender, EventArgs e)	Відкриття диска за допомогою лівої кнопки миші
bool OpenFile(List<string> Dir, string FileName)	Реалізація відкриття файлу
private void CopyFile(object sender, EventArgs e)	Реалізація копіювання файлу
private void InsertFile(object sender, EventArgs e)	Реалізація вставлення файлу
private void DeleteFile(object sender, EventArgs e)	Реалізація видалення файлу
private void CreateTextFile(object sender, EventArgs e)	Реалізація створення текстового файлу
private void CreateFolder(object sender, EventArgs e)	Реалізація створення папки
private void CreateBmpFile(object sender, EventArgs e)	Реалізація створення точкового малюнку

Продовження табл. 3.2.

1	2
private void CreateDocxFile(object sender, EventArgs e)	Реалізація створення документу
private void CreateXlsxFile(object sender, EventArgs e)	Реалізація створення електронної таблиці
private void CreatePptxFile(object sender, EventArgs e)	Реалізація створення презентації
private void CutFile(object sender, EventArgs e)	Реалізація вирізання файлу
private void RenameFile(object sender, EventArgs e)	Реалізація перейменування файлу
private void PropertiesFile(object sender, EventArgs e)	Реалізація виведення інформації про файл або папку
string BuildPath(List<string> Dir, string FileName)	Будування шляху до файлу
string BuildPath(List<string> Dir)	Будування шляху до папки
void CopyDir(string FromDir, string ToDir)	Рекурсивна функція для копіювання змісту папки
bool Create402(string OldName, string Type)	Створення файлу при помилці 402
private void DiskContextMenu_Opening(object sender, CancelEventArgs e)	Відкриття випадаючого меню для дисків
private void DiskInfo_Click(object sender, EventArgs e)	Відкриття форми властивостей диску
private void FileContextMenuStrip_Opening(object sender, CancelEventArgs e)	Відкриття випадаючого меню для файлів та папок

Продовження табл. 3.2.

1	2
private void HelpToolStripMenuItem_Click(object sender, EventArgs e)	Відкриття форми довідка
private void ToolTipLoad()	Реалізація підказок
private void About(object sender, EventArgs e)	Відкриття форми з інформацією про програму
private void StandartRadioButton_CheckedChanged(object sender, EventArgs e)	Вибір стандартної теми
private void BlackRadioButton_CheckedChanged(object sender, EventArgs e)	Вибір темної теми
private void LightRadioButton_CheckedChanged(object sender, EventArgs e)	Вибір світлої теми
private void UkrainianRadioButton_CheckedChanged(object sender, EventArgs e)	Вибір української мови
private void EnglishRadioButton_CheckedChanged(object sender, EventArgs e)	Вибір англійської мови
void SetStyle()	Налаштування форми під обрану тему
void SetLanguage()	Налаштування форми під обрану мову
<i>public:</i>	
public MainForm()	Конструктор, який налаштовує форму при першому запуску

Продовження табл.3.2.

1	2
public string NewName	Дозволяє отримувати та змінювати змінну newName
public string Name402	Дозволяє отримувати та змінювати змінну name402

3.4.2 Клас AboutForm

Ціль класу полягає у відображенні інформації про програмне забезпечення за допомогою форми. Дані та методи класу наведені у табл. 3.3.

Таблиця 3.3 – Опис полів та методів класу AboutForm

Поля та методи класу	Опис
<i>public:</i>	
public string FormStyle	Дозволяє отримувати та змінювати зміну formStyle
public string FormLanguage	Дозволяє отримувати та змінювати зміну formLanguage
public AboutForm()	Конструктор форми
<i>private:</i>	
string formStyle	Змінна для збереження поточної теми
string formLanguage	Змінна для збереження поточної мови
private void AboutForm_Load(object sender, EventArgs e)	Налаштування форми при її відкритті
void SetStyle()	Налаштування форми під обрану тему
void SetLanguage()	Налаштування форми під обрану мову

3.4.3 Клас DiskInfoForm

Ціль класу полягає у відображенні необхідної інформації про диск за допомогою форми. Дані та методи класу наведені у табл. 3.4.

Таблиця 3.4 – Опис полів та методів класу DiskInfoForm

Поля та методи класу	Опис
<i>public:</i>	
public string FormStyle	Дозволяє отримувати та змінювати зміну formStyle
public string FormLanguage	Дозволяє отримувати та змінювати зміну formLanguage
public DiskInfoForm(DiskObject SelectedDisk)	Конструктор форми
<i>private:</i>	
string formStyle	Змінна для збереження поточної теми
string formLanguage	Змінна для збереження поточної мови
DiskObject SelectedDisk	Змінна для збереження інформації про обраний диск
private void DiskInfoForm_Load(object sender, EventArgs e)	Налаштування форми при її відкритті
void SetStyle()	Налаштування форми під обрану тему
void SetData()	Налаштування форми під обрану мову та встановлення інформації в форму

3.4.4 Клас ErrorForm

Ціль класу полягає у відображенні виняткової ситуації за допомогою форми. Дані та методи класу наведені у табл. 3.5.

Таблиця 3.5 – Опис полів та методів класу ErrorForm

Поля та методи класу	Опис
<i>public:</i>	
public string FormStyle	Дозволяє отримувати та змінювати зміну formStyle
public string FormLanguage	Дозволяє отримувати та змінювати зміну formLanguage
public int Code	Дозволяє отримувати та змінювати зміну code
public ErrorForm()	Конструктор форми
<i>private:</i>	
string formStyle	Змінна для збереження поточної теми
string formLanguage	Змінна для збереження поточної мови
int code	Змінна для збереження коду помилки
private void ErrorForm_Load(object sender, System.EventArgs e)	Налаштування форми при її відкритті
void SetStyle()	Налаштування форми під обрану тему
void SetData()	Налаштування форми під обрану мову та встановлення інформації в форму
void UkrainianErrorText()	Існуючі помилки українською мовою
void EnglishErrorText()	Існуючі помилки англійською мовою
private void OkButton_Click(object sender, System.EventArgs e)	Реалізація реакції на натискання кнопки «ОК»

3.4.5 Клас FilePropertiesForm

Ціль класу полягає у відображенні інформації про файл за допомогою форми. Дані та методи класу наведені у табл. 3.6.

Таблиця 3.6 – Опис полів та методів класу FilePropertiesForm

Поля та методи класу	Опис
1	2
<i>public:</i>	
public string FormStyle	Дозволяє отримувати та змінювати зміну formStyle
public string FormLanguage	Дозволяє отримувати та змінювати зміну formLanguage
public string Path	Дозволяє отримувати та змінювати зміну path
public string Type	Дозволяє отримувати та змінювати зміну type
public FilePropertiesForm()	Конструктор форми
public static long DirSize(DirectoryInfo d)	Підрахунок розміру папки
<i>private:</i>	
string formStyle	Змінна для збереження поточної теми
string formLanguage	Змінна для збереження поточної мови
string path	Змінна для збереження інформації про шлях до файлу чи папки
string type	Змінна для збереження інформації про тип файлу чи папки
private void FilePropertiesForm_Load(object sender, EventArgs e)	Налаштування форми при її відкритті

Продовження табл. 3.6

1	2
void SetStyle()	Налаштування форми під обрану тему
void SetData()	Встановлення інформації в форму
void SetLanguage()	Налаштування форми під обрану мову

3.4.6 Клас RenamingForm

Ціль класу полягає у реалізації зміни назви папки або файлу за допомогою форми. Дані та методи класу наведені у табл. 3.7.

Таблиця 3.7 – Опис полів та методів класу RenamingForm

Поля та методи класу	Опис
1	2
<i>public:</i>	
public string FormStyle	Дозволяє отримувати та змінювати зміну formStyle
public string FormLanguage	Дозволяє отримувати та змінювати зміну formLanguage
public MainForm Main	Дозволяє отримувати та змінювати зміну main
public string OldName	Дозволяє отримувати та змінювати зміну oldName
public bool Error402	Дозволяє отримувати та змінювати зміну error402
public RenamingForm()	Конструктор форми
<i>private:</i>	
string formStyle	Змінна для збереження поточної теми
string formLanguage	Змінна для збереження поточної мови

Продовження табл. 3.7.

1	2
MainForm main	Змінна для посилання на головну форму

string oldName	Змінна для збереження старої назви файлу чи папки
bool error402	Змінна, яка вказує чи відбувається перейменування через помилку
private void RenamingForm_Load(object sender, EventArgs e)	Налаштування форми при її відкритті
void SetStyle()	Налаштування форми під обрану тему
void SetLanguage()	Налаштування форми під обрану мову
private void ConfirmButton_Click(object sender, EventArgs e)	Реалізація реакції на натискання кнопки «Підтвердити»
private void CancelButton_Click(object sender, EventArgs e)	Реалізація реакції на натискання кнопки «Відмінити»

3.4.7 Клас HelpForm

Ціль класу полягає у відображенні інформації про програмне забезпечення за допомогою форми. Дані та методи класу наведені у табл.3.8.

Таблиця 3.8 – Опис полів та методів класу HelpForm

Поля та методи класу	Опис
1	2
<i>public:</i>	
public HelpForm()	Конструктор форми

Продовження табл. 3.8.

1	2
public string FormLanguage	Дозволяє отримувати та змінювати зміну formLanguage

public string FormStyle	Дозволяє отримувати та змінювати зміну formStyle
<i>private:</i>	
string formStyle	Змінна для збереження поточної теми
string formLanguage	Змінна для збереження поточної мови
string HelpTextUkrainian	Змінна для збереження тексту довідки українською мові
string HelpTextEnglish	Змінна для збереження тексту довідки англійською мові
private void HelpForm_Load(object sender, EventArgs e)	Налаштування форми при її відкритті
void SetStyle()	Налаштування форми під обрану тему
void SetLanguage()	Налаштування форми під обрану мову

3.4.8 Клас FileObject

Ціль класу полягає у збереженні інформації про фали. Дані та методи класу наведені у таблиці 3.9.

Таблиця 3.9 – Опис полів та методів класу HelpForm

Поля та методи класу	Опис
1	2
<i>public:</i>	
public string Size	Дозволяє отримувати та змінювати зміну size

Продовження табл. 3.9.

1	2
public string Name	Дозволяє отримувати та змінювати зміну name

public string Type	Дозволяє отримувати та змінювати зміну type
internal DataAndTime ChangeDate	Дозволяє отримувати та змінювати зміну changeDate
<i>private:</i>	
DataAndTime changeDate	Змінна для збереження дати та часу останньої зміни файлу
string size	Змінна для збереження розміру
string name	Змінна для збереження назви файлу
string type	Змінна для збереження типу файлу

3.4.9 Клас Style

Ціль класу полягає у налаштуванні та збереженні палітри кольорів відповідно до вибраної теми. Дані та методи класу наведені у табл. 3.10.

Таблиця 3.10 – Опис полів та методів класу Style

Поля та методи класу	Опис
1	2
<i>public:</i>	
public Color MainColor	Дозволяє отримувати та змінювати зміну mainColor
public Color SubColor	Дозволяє отримувати та змінювати зміну subColor

Продовження табл. 3.10.

1	2
public Color BackColor	Дозволяє отримувати та змінювати зміну backColor

public Color ForeColor	Дозволяє отримувати та змінювати зміну foreColor
public Color ThirdColor	Дозволяє отримувати та змінювати зміну thirdColor
public Color FreeColor	Дозволяє отримувати та змінювати зміну freeColor
public Color UsedColor	Дозволяє отримувати та змінювати зміну usedColor
public ChartColorPalette Palette	Дозволяє отримувати та змінювати зміну palette
public int CornerRadius	Дозволяє отримувати та змінювати зміну cornerRadius
public ToolStripProfessionalRenderer Renderer	Дозволяє отримувати та змінювати зміну foreColor
public void SetStyleSetting(string StyleSetting)	Функція, яка встановлює стиль
<i>private:</i>	
Color mainColor	Змінна для збереження головного кольору теми
Color subColor	Змінна для збереження вторинного кольору теми
Color thirdColor	Змінна для збереження третього кольору теми
Color backColor	Змінна для збереження кольору фону

Продовження табл. 3.10

1	2
Color foreColor	Змінна для збереження кольору шрифту та границь елементів

Color freeColor	Змінна для збереження кольору на діаграмі відповідного до вільного місця
Color usedColor	Змінна для збереження кольору на діаграмі відповідного до зайнятого місця
ChartColorPalette palette	Змінна для збереження палітри кольорів діаграми
int cornerRadius	Змінна для збереження радіуси границі елементів
ToolStripProfessionalRenderer renderer	Змінна для збереження відмальовувача елементу MenuStrip

3.4.10 Клас ToolStripRendererStandart

Ціль класу полягає у відмальовуванні елементу ToolStrip у стандартній темі. Дані та методи класу наведені у табл. 3.11.

Таблиця 3.11 – Опис полів та методів класу ToolStripRendererStandart

Поля та методи класу	Опис
<i>public:</i>	
public ToolStripRendererStandart() : base(new ColsStandart())	Встановлює налаштування відмальовування елементу ToolStrip у стандартній темі

3.4.11 Клас ColsStandart

Ціль класу полягає у зберіганні налаштувань відмальовування елементу ToolStrip у стандартній темі. Дані та методи класу наведені у табл.3.12.

Таблиця 3.12 – Опис полів та методів класу ColsStandart

Поля та методи класу	Опис
<i>public:</i>	
public override Color MenuItemSelected	Встановлює колір виділеного елемента меню
public override Color ToolStripDropDownBackground	Встановлює колір фону меню
public override Color ImageMarginGradientEnd	Встановлює колір початку градієнту меню
public override Color ImageMarginGradientMiddle	Встановлює колір середини градієнту меню
public override Color MenuItemSelectedGradientBegin	Встановлює колір початку градієнту виділеного елемента меню
public override Color MenuItemSelectedGradientEnd	Встановлює колір кінця градієнту виділеного елемента меню
public override Color MenuItemPressedGradientBegin	Встановлює колір початку градієнту натиснутого елемента меню
public override Color MenuItemPressedGradientMiddle	Встановлює колір середини градієнту натиснутого елемента меню
public override Color MenuItemPressedGradientEnd	Встановлює колір кінця градієнту натиснутого елемента меню
public override Color MenuItemBorder	Встановлює колір границь елементів меню

3.4.12 Клас MenuStripRendererBlack

Ціль класу полягає у відмальовуванні елементу MenuStrip у темній темі. Дані та методи класу наведені у табл. 3.13.

Таблиця 3.13 – Опис полів та методів класу MenuStripRendererBlack

Поля та методи класу	Опис
<i>public:</i>	
<code>public MenuStripRendererBlack() : base(new ColsBlack())</code>	Встановлює налаштування відмальовування елементу MenuStrip у темній темі

3.4.13 Клас ColsBlack

Ціль класу полягає у зберіганні налаштувань відмальовування елементу MenuStrip у темній темі. Дані та методи класу наведені у табл. 3.14.

Таблиця 3.14 – Опис полів та методів класу ColsBlack

Поля та методи класу	Опис
1	2
<i>public:</i>	
<code>public override Color MenuItemSelected</code>	Встановлює колір виділеного елемента меню
<code>public override Color ToolStripDropDownBackground</code>	Встановлює колір фону меню
<code>public override Color ImageMarginGradientEnd</code>	Встановлює колір початку градієнту меню
<code>public override Color ImageMarginGradientMiddle</code>	Встановлює колір середини градієнту меню

Продовження табл. 3.14.

1	2
<code>public override Color MenuItemSelectedGradientBegin</code>	Встановлює колір початку градієнту виділеного елемента меню

public	override	Color	Встановлює колір кінця градієнту виділеного елемента меню
MenuItemSelectedGradientEnd			
public	override	Color	Встановлює колір початку градієнту натиснутого елемента меню
MenuItemPressedGradientBegin			
public	override	Color	Встановлює колір середини градієнту натиснутого елемента меню
MenuItemPressedGradientMiddle			
public	override	Color	Встановлює колір кінця градієнту натиснутого елемента меню
MenuItemPressedGradientEnd			
public	override	Color	Встановлює колір границь елементів меню
MenuItemBorder			

3.4.14 Клас MenuStripRendererLight

Ціль класу полягає у відмальовуванні елемента MenuStrip у світлій темі. Дані та методи класу наведені у табл.3.15.

Таблиця 3.15 – Опис полів та методів класу MenuStripRendererLight

Поля та методи класу	Опис
<i>public:</i>	
public MenuStripRendererLight() : base(new ColsLight())	Встановлює налаштування відмальовування елемента MenuStrip у світлій темі

3.4.15 Клас ColsLight

Ціль класу полягає у зберіганні налаштувань відмальовування елемента MenuStrip у світлій темі. Дані та методи класу наведені у табл. 3.16.

Таблиця 3.16 – Опис полів та методів класу ColsLight

Поля та методи класу	Опис
<i>public:</i>	
public override Color MenuItemSelected	Встановлює колір виділеного елемента меню
public override Color ToolStripDropDownBackground	Встановлює колір фону меню
public override Color ImageMarginGradientEnd	Встановлює колір початку градієнту меню
public override Color ImageMarginGradientMiddle	Встановлює колір середини градієнту меню
public override Color MenuItemSelectedGradientBegin	Встановлює колір початку градієнту виділеного елемента меню
public override Color MenuItemSelectedGradientEnd	Встановлює колір кінця градієнту виділеного елемента меню
public override Color MenuItemPressedGradientBegin	Встановлює колір початку градієнту натиснутого елемента меню
public override Color MenuItemPressedGradientMiddle	Встановлює колір середини градієнту натиснутого елемента меню
public override Color MenuItemPressedGradientEnd	Встановлює колір кінця градієнту натиснутого елемента меню
public override Color MenuItemBorder	Встановлює колір границь елементів меню

3.5 Основні рішення щодо розробки інтерфейсу

Бачення на вигляд інтерфейсу програмного застосунку у користувачів може бути різне. Саме через різні погляди на дизайн, приємну кольорову гаму, було вирішено створити декілька тем, а саме: стандартна; темна та світла.

В стандартній темі переважають сірі відтінки тому, що більшість людей можуть відрізняти сірі відтінки (наприклад дальтоніки), а також ці кольори є спокійними та не навантажують око людини. Сірий підходить людям з високою швидкістю життя. Він загальмовує нервову систему та заспокоює. Колір впливає на роботу мозку, допомагає поглянути на проблему без емоцій, із ясною головою. Кольорова гама цієї теми зображена на рис.3.2.

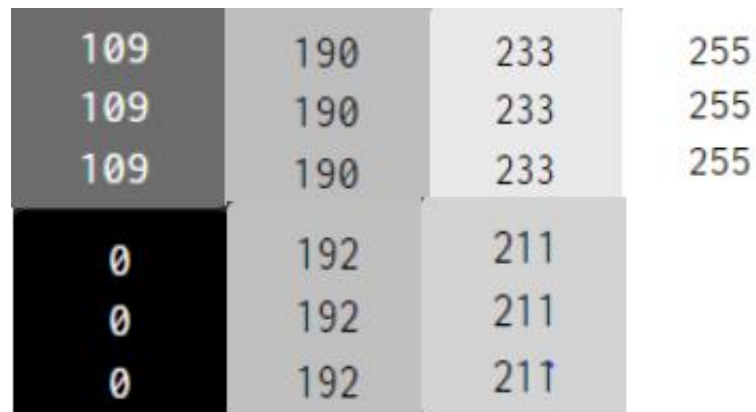


Рисунок 3.2 – Кольорова гама теми «Стандарт»

Приклад коду для оголошення змінних теми «Стандарт»:

```
case "Standart":
    MainColor = Color.FromArgb(109, 109, 109);
    SubColor = Color.FromArgb(190, 190, 190);
    BackColor = Color.FromArgb(233, 233, 233);
    ThirdColor = Color.White;
    ForeColor = Color.Black;
    FreeColor = Color.LightGray;
    UsedColor = Color.Silver;
    Palette = ChartColorPalette.Grayscale;
    CornerRadius = 15;
    Renderer = new MenuStripRendererStandart();
    break;
```

Деяким користувачам сірий колір може здаватися нудним, але якісь яскраві кольори будуть занадто привертати увагу. Тому була створена темна тема. Кольорова гама цієї теми зображена на рис. 3.3.

0	46	0	255
0	139	255	255
255	87	0	255
31	61	102	
48	94	205	
78	154	170	

Рисунок 3.3 – Кольорова гама теми «Темна»

Приклад коду для оголошення змінних теми «Темна»:

```
case "Black":
    MainColor = Color.Blue;
    SubColor = Color.Green;
    BackColor = Color.FromArgb(31, 48, 78);
    ThirdColor = Color.FromArgb(61, 94, 154);
    ForeColor = Color.White;
    FreeColor = Color.SeaGreen;
    UsedColor = Color.MediumAquamarine;
    Palette = ChartColorPalette.SeaGreen;
    CornerRadius = 15;
    Renderer = new MenuStripRendererBlack();
    break;
```

Здається, що темні кольори мають складний характер, але насправді темна фонові гама дуже добре поєднується з іншими палітрами. І в цьому її безперечна перевага. Саме тому в темній темі були використані насичені кольори, такі як зелений та синій. Насичені відтінки здатні неймовірним чином об'єднати, гармонізувати та впорядкувати чималу кількість об'єктів з темним кольором.

Світлі відтінки не позбавлені своєї індивідуальності. Вони можуть бути рожевими, зеленими, бузковими. Людина розрізняє до 100 000 відтінків і 1/3 з них світлі. Говорять що світлі кольори – психологічно комфортні, а особливо теплі відтінки. Також вони поєднуються з багатьма іншими кольорами. Кольорова гама цієї теми зображена на рис. 3.4.

255	255	255	255
174	209	255	255
66	145	223	255
0	30	255	
0	144	192	
0	255	203	

Рисунок 3.4 – Кольорова гама теми «Світла»

Приклад коду для оголошення змінних теми «Світла»:

```
case "Light":
    MainColor = Color.FromArgb(255, 209, 145);
    SubColor = Color.Pink;
    BackColor = Color.FromArgb(255, 255, 223);
    ThirdColor = Color.White;
    ForeColor = Color.Black;
    FreeColor = Color.DodgerBlue;
    UsedColor = Color.FromArgb(255, 174, 66);
    Palette = ChartColorPalette.BrightPastel;
    CornerRadius = 15;
    Renderer = new MenuStripRendererLight();
    break;
```

Закруглені елементи є легшими для сприйняття. Закругленість виглядає сучасно. Застосування повного закруглення почалося в мобільних інтерфейсах і перекинувало до програм на персональному комп'ютері. Відсутність гостроти викликає почуття простоти, оптимізму та доступності. Ймовірно, саме тому багато систем дизайну взяли закруглені елементи на озброєння і широко застосовують для іконок, кнопок або навіть зображень (наприклад новий інтерфейс в Windows 11). В створеному інтерфейсі також були використані закруглені елементи, а саме: кнопки та текстові поля.

Приклад коду для закруглення та кольорової палітри для кнопок:

```
OkButton.FillColor = StyleSetter.MainColor;
OkButton.FillColor2 = StyleSetter.SubColor;
OkButton.BackColor = StyleSetter.BackColor;
OkButton.BorderRadius = StyleSetter.CornerRadius;
OkButton.ForeColor = StyleSetter.ForeColor;
```

Для налаштування та збереження палітри кольорів відповідно до вибраної теми було створено клас `Style`. Також кожна форма має власну функцію `void SetStyle()`, де відбувається налаштування самої форми під обрану тему.

Було розроблені зображення для різних типів файлів та папок, які представлено на рис. 3.5



Рисунок 3.5 – Створені зображення для файлів та папок

Приклад коду для завантаження зображення для текстового файлу

```
if (Info.Extension == ".txt")
{
    FilePictureBox.Image
Image.FromFile(@"..\images\bloknot.png");
}
```

Як і кожний застосунок, програмне забезпечення «FreeFile» має власну іконку, яке зображено на рис. 3.6

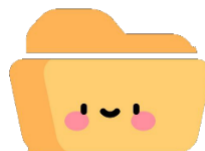


Рисунок 3.6 – Іконка файлового менеджера «FreeFile»

Для більшої гнучкості інтерфейс має українську та англійську локалізацію, яку зручно можна обрати на головній формі (див. рис. 3.7). Також там можна й обрати потрібну тему.

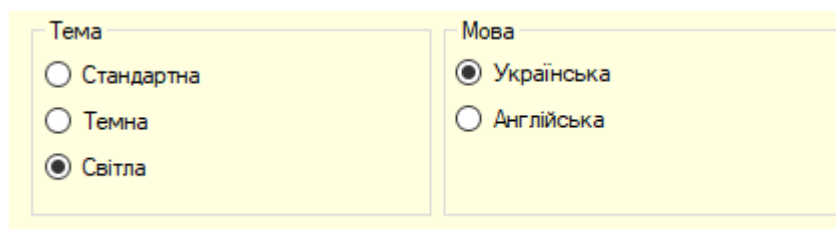


Рисунок 3.7 – Налаштування інтерфейсу

Приклад коду для реалізації переключення на тему «Стандарт»:

```
private void StandartRadioButton_CheckedChanged(object sender,
EventArgs e)
{
    if (StandartRadioButton.Checked)
    {
        FormStyle = "Standart";
        SetStyle();
    }
}
```

Розроблено наступні форми програмного забезпечення «FreeFile»:

- «Головна форма» – дозволяє виконувати операції з файлами, папками та дисками, переглядати обрану директорію, шлях до неї, а також змінювати поточну директорію;
- «Справка» – містить інформацію про основний функціонал ПЗ;
- «Про програму» – містить інформацію про програму та її розробника;
- «Властивості файлу або папки» – містить інформацію про обраний файл або папку (назва, розширення, шлях, розмір та дата останньої зміни);
- «Інформація про диск» – містить інформацію про обраний диск (назва, скільки є місця взагалі, скільки зайнято та вільно);
- «Переименовання» – реалізує перейменування файлу або папки;
- «Помилка» – реалізує виведення інформацію про певну виняткову ситуацію, яка сталась в програмі.

Структура інтерфейсу системи наведена на рис. 3.7



Рисунок 3.7 – Структура інтерфейсу системи

Код головного файлу проєкту наведено в Додатку А. Інші файли не наводяться у зв'язку з обмеженнями до об'єму пояснювальної записки.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

В даному розділі розглянуто призначення програми, умови її застосування, функції, характеристика, структура, засоби звертання вхідні та вихідні дані, а також повідомлення які може відобразити застосунок.

4.1 Призначення та умови застосування програми

Програма призначена для надання користувачу зручного інтерфейсу для організації управління файловою системою та файлами.

4.2 Характеристики програми

Функції, за допомогою яких користувач може управляти файловою системою ОС Windows 10 описані у п. 1.2.3. Вимоги до технічних засобів та операційної системи наведені у п. 1.2.4.

Для запуску програмного забезпечення необхідно виконати файл FreeFile.exe. Програма створена мовою C# у середовищі Visual Studio 2019 на платформі .NET Framework 4.7.2. Застосунок має головне вікно на якому відображено вміст поточної директорії, шлях до неї, підключені диски. Також на ньому можна виконувати операції з файлами, папками та дисками, змінювати тему та мову інтерфейсу, відкрити інформаційні вікна.

4.3 Структура програми

При початку роботи програми завантажується головне вікно. Разом з ним створюється і запускається пакетний файл (файл dir.bat), який отримує інформацію про поточну директорію (при запуску це диск C) та записує її у текстовий файл (файл dir.txt).

Також виконується консольна команда для отримання інформації про підключені носії даних у текстовий файл (файл disk.txt). При помилці у цих процесах виникне вікно, яке повідомить про це. Далі інформація з створених текстових файлів буде виведена на форму.

Загальна структура програми наведена на рис. 4.1 та складається з наступних компонентів: файлів бібліотек, програмного коду, форм.

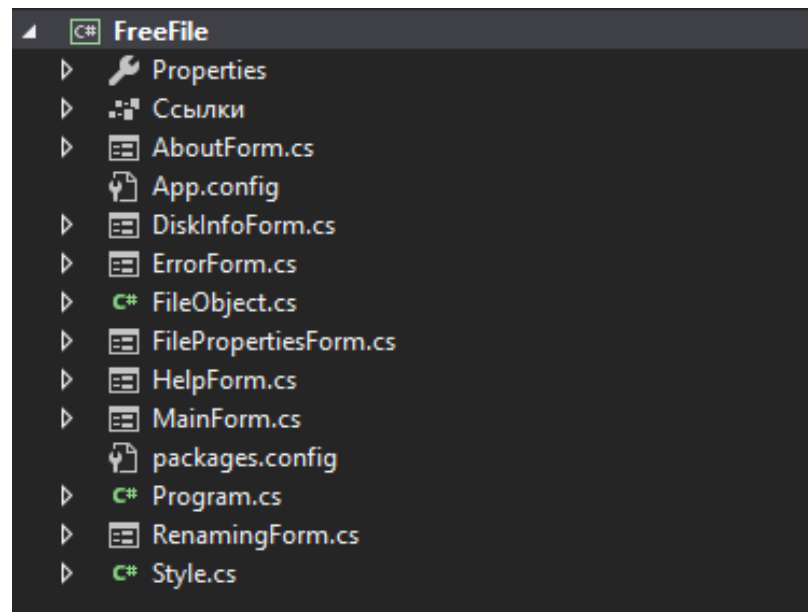


Рисунок 4.1 – Структура програми

Програма має такі методи звертання до неї:

- ліва (взаємодія з дисками, файлами та папками, системним меню, кнопками) та права (відкриття спливаючих меню) клавіші миші;
- гарячі клавіші (операції з файлами);
- клавіші клавіатури (ввід назви файлу або папки при перейменуванні).

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Програма отримує такі вхідні дані як:

- сигнал про натискання клавіш миші та клавіатури;
- індекс обраного користувачем елементу списку;
- інформація про обрані перемикачі теми та мови інтерфейсу;
- інформація про підключені носії даних (файл disk.txt);
- інформація про зміст поточної директорії (файл dir.txt);
- зображення іконки програми, дисків, папок та файлів.

4.4.2 Вихідні дані

Програма видає такі вихідні дані як:

- інформація про підключені носії даних;
- шлях до поточної директорії;
- інформація про зміст поточної директорії;
- спливаючі підказки;
- інформація про програму та розробника;
- інформація про основний функціонал ПЗ;
- пакетний файл з консольними командами (файл dir.bat).

4.5 Повідомлення програмісту

Програма містить такі повідомлення як:

- «401 Не введено назву файлу» – означає, що при перейменуванні не введено нову назву файлу;
- «402 Файл з такою назвою вже існує» – означає, що файл з такою назвою та розширенням вже є, тому програма не може створити ще один такий;
- «501 Відсутній доступ до файлу» – означає, що програма не може отримати доступ до файлу, бо він зайнятий іншим процесом;
- «502 Некоректна побудова шляху» – означає, що шлях до файлу був побудований некоректно і не може бути використаним;

- «503 Невдале завантаження даних» – означає, що програма не може коректно завантажити данні з файлу;
- «504 Помилка відкриття файлу» – означає, що програма не може відкрити файл;
- «505 Невдала операція з файлом» – означає, що операція з файлом пройшла некоректно.

5 КЕРІВНИЦТВО ОПЕРАТОРА

В даному розділі розглянуто призначення програми, умови її виконання, процес виконання програми та повідомлення для користувача.

5.1 Призначення програми

Програма призначена для надання користувачу зручного інтерфейсу для організації роботи з файловою системою та файлами.

5.2 Умови виконання програми

Для забезпечення нормальної роботи застосунку необхідно склад апаратно-технічних засобів наведений у п. 1.2.4.

Для комфортної роботи з програмою від користувача потрібні:

- знання та володіння українською або англійською мовою;
- навички користування комп'ютерною мишею та клавіатурою;
- ознайомлення з довідковою інформацією щодо користування програмою.

5.3 Виконання програми

Програма запускається за допомогою виконуваного файлу FreeFile.exe шляхом подвійного натискання лівої клавіші комп'ютерної миші по ньому або його ярлику.

При запуску програми користувачу відображається головне вікно, яке зображено на рис. 5.1.

На рис. 5.1 прямокутником червоного кольору виділено список з підключеними дисками, прямокутником зеленого – вміст поточної директорії, прямокутником синього – шлях до поточної директорії, прямокутником

жовтого — налаштування інтерфейсу: обрання теми та мови. Приклад налаштованого інтерфейсу під бажання користувачів наведено на рис. 5.2.

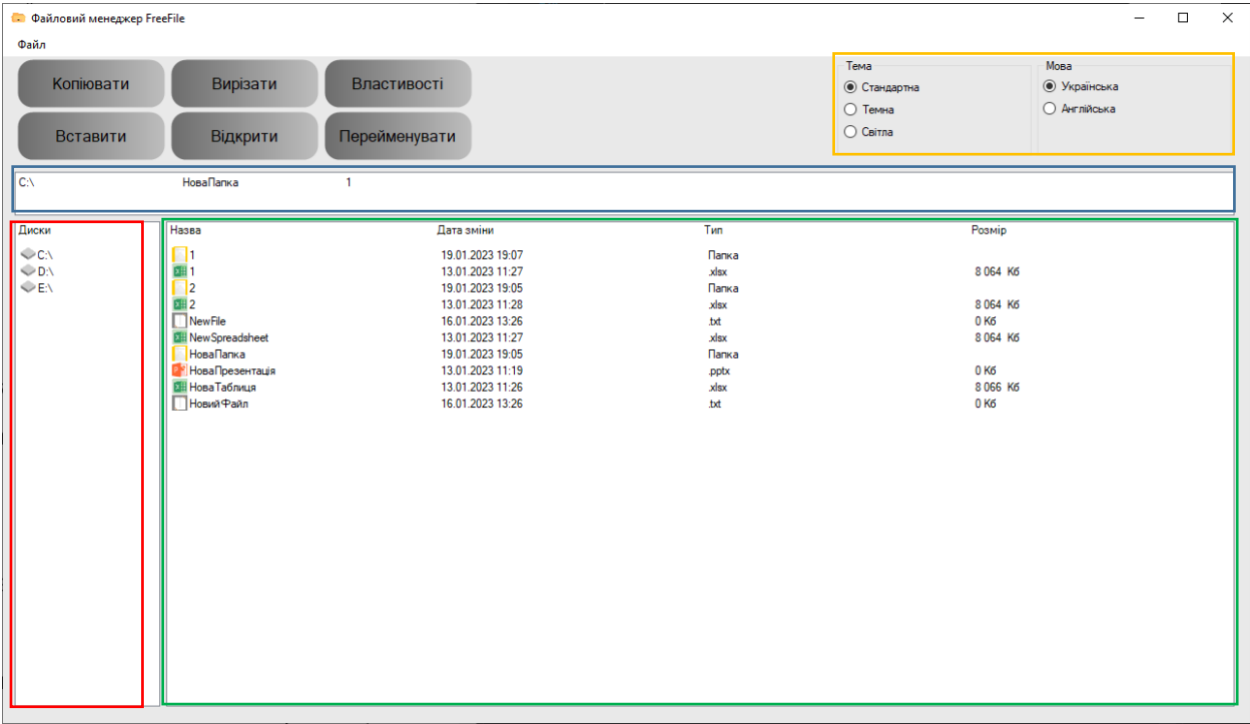


Рисунок 5.1 – Головне вікно

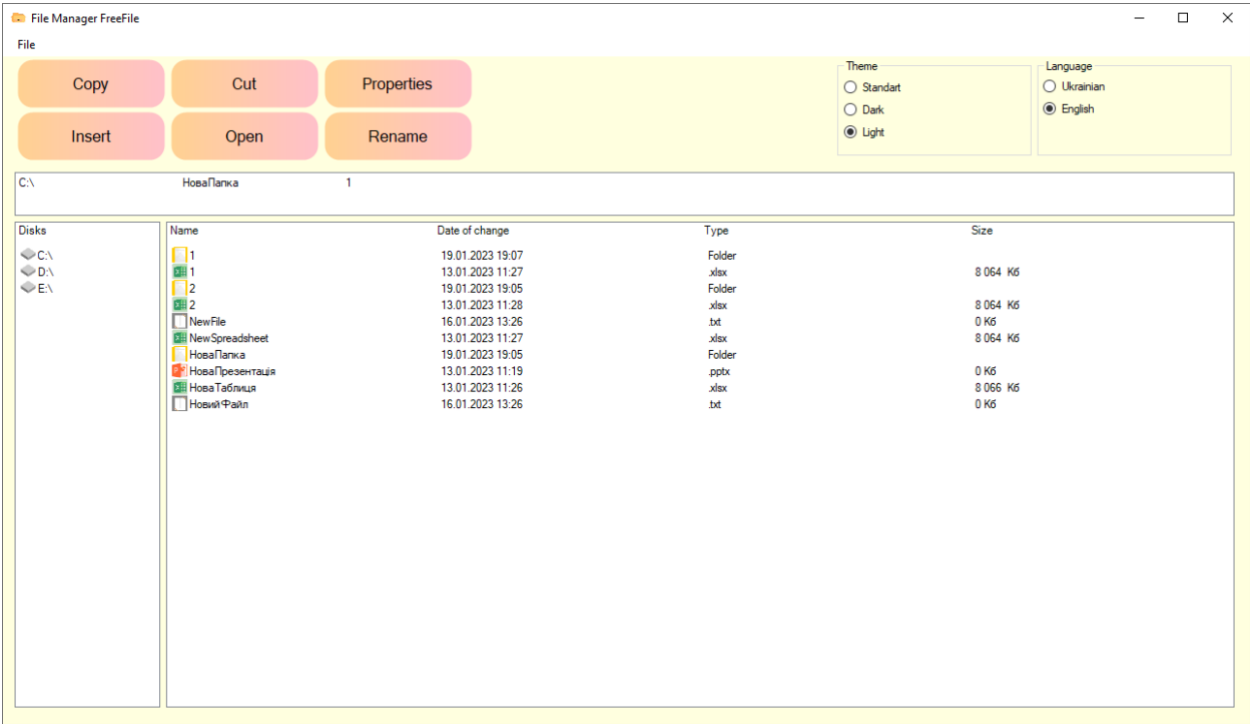


Рисунок 5.2 – Приклад налаштованого інтерфейсу

Для того щоб виконати операцію з файлом чи папкою треба натиснути на виділений файл або папку правою кнопкою миші. Після цього з спливаючого меню, яке зображено на рис. 5.3, можна обрати потрібну операцію

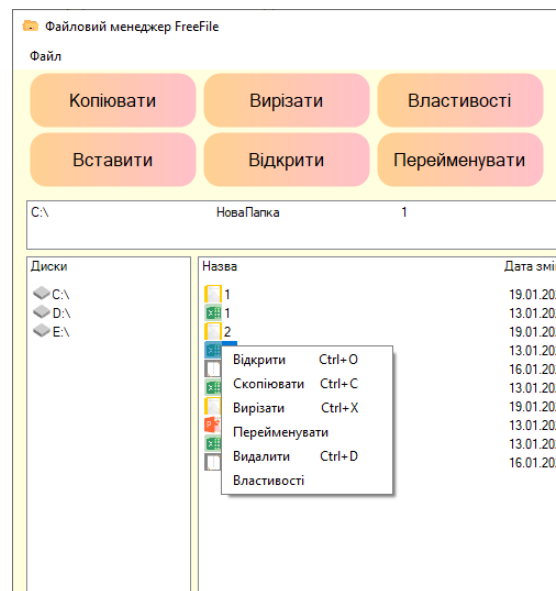


Рисунок 5.3 – Меню для віділеного файлу чи папки

Також операцію з файлом можна провести за допомогою гарячих клавіш та кнопок на головному вікні. Як приклад розглянемо операцію «Перейменування». У спливаючому меню, яке зображено на рис. 5.3, потрібно обрати пункт «Перейменування». Після чого відкривається вікно, яке продемонстровано на рис. 5.4.

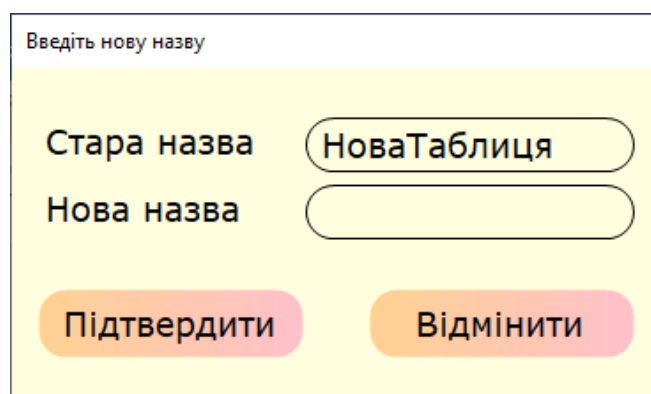


Рисунок 5.4 – Вікно перейменування

Тут користувачу необхідно ввести нову назву та натиснути кнопку «Підтвердити». Якщо нова назва співпадає зі старою назвою файлу, то відобразиться сповіщення про помилку, яке показано на рис. 5.5.

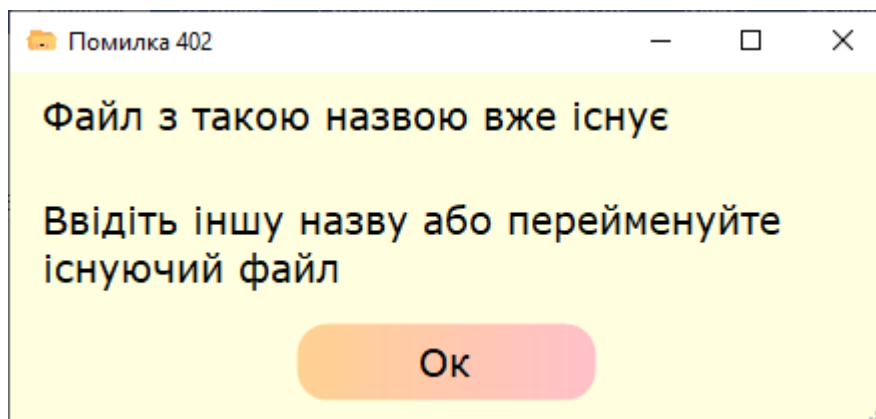


Рисунок 5.5 – Сповіщення про помилку

При натисканні на кнопку «Відмінити» (рис. 5.4) вікно «Перейменування» буде закрито.

Якщо необхідно дізнатися інформацію про файл або папку, то у спливаючому меню, яке зображено на рис. 5.3, потрібно обрати пункт «Властивості». Після чого відкривається вікно, яке наведено на рис. 5.6.

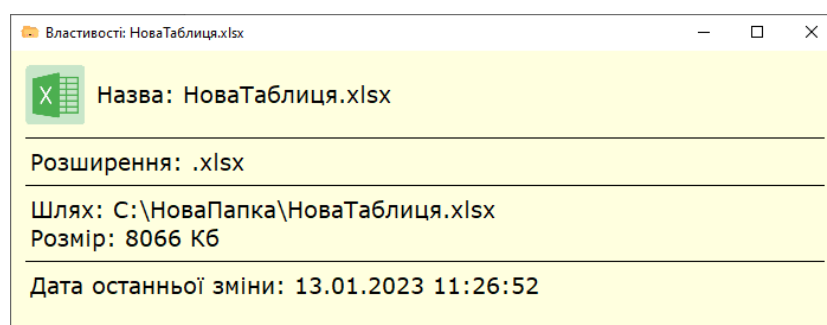


Рисунок 5.6 – Властивості обраного файлу або папки

При натисканні на пусте місце правою кнопкою миші на головному вікні, яке зображено на рис. 5.1, у списку вмісту поточної директорії, буде відображено спливаюче меню, яке зображено на рис. 5.7.

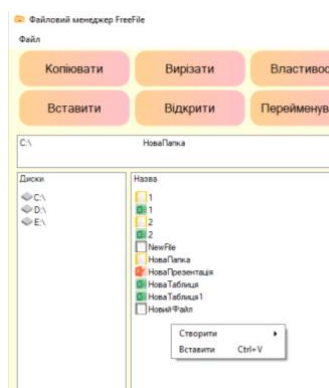


Рисунок 5.7 – Меню після натискання на пусте місце

У випадках створення нового файлу чи папки необхідно відкривати меню, яке зображено на рис. 5.7, і після чого у підпунктах для створення обирати необхідний тип файлу чи папки. Якщо при створенні вже є файл цього ж розширення з таким же ім'ям, то відобразиться сповіщення про помилку, яке продемонстровано на рис. 5.5. Після чого користувачу у вікно, що зображено на рис. 5.4, потрібно ввести нову назву, яка буде відрізнятися від попередньої.

Якщо натиснути правою кнопкою миші на виділений диск на головній формі (рис. 5.1), то відкривається спливаюче меню для дисків, що представлено на рис. 5.8.

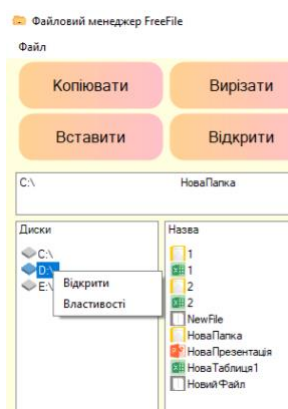


Рисунок 5.8 – Меню для віділеного диску

При натисканні на пункт «Властивості» у меню для виділеного диску, відкривається вікно «Інформація про диск», яке показано на рис. 5.9.

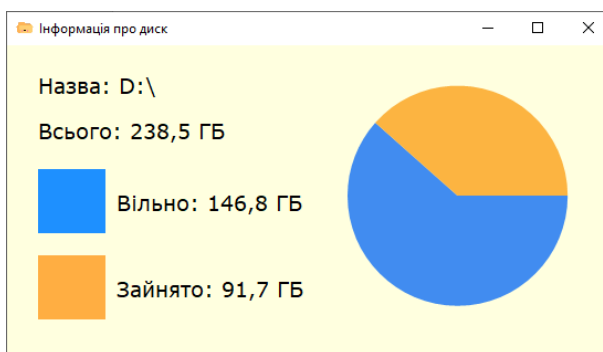


Рисунок 5.9 – Вікно з інформацією про диск

Для отримання довідкової інформації є вікна «Справка» та «Про програму». Щоб їх відкрити, треба обрати необхідний пункт у меню «Файл», що показано на рис. 5.10

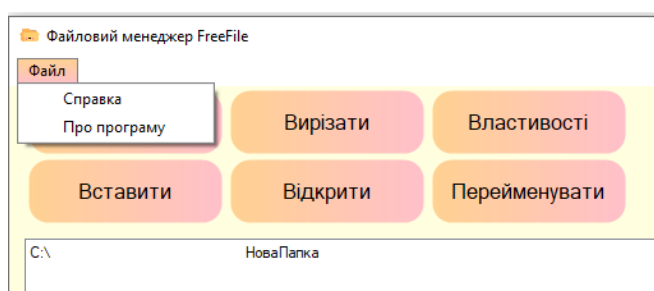


Рисунок 5.10 – Меню «Файл»

5.4 Повідомлення оператора

Програма містить такі повідомлення як:

- «Не введено назву файлу» – означає, що при перейменуванні не введено нову назву файлу;
- «Файл з такою назвою вже існує» – означає, що файл з такою назвою та розширенням вже є, тому програма не може створити ще один такий;
- «Відсутній доступ до файлу» – означає, що програма не може отримати доступ до файлу, бо він зайнятий іншим процесом;
- «Некоректна побудова шляху» – означає, що шлях до файлу був побудований некоректно і не може бути використаним;

- «Невдале завантаження даних» – означає, що програма не може коректно завантажити данні з файлу;
- «Помилка відкриття файлу» – означає, що програма не може відкрити файл;
- «Невдала операція з файлом» – означає, що операція з файлом пройшла некоректно.

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи бакалавра вирішено актуальну задачу розробки програмного застосунку, що забезпечує інтерактивну взаємодію користувача з файловою системою ОС Windows 10.

Для досягнення мети проекту було проведено ретельний аналіз області застосування, а також огляд існуючих програмних рішень. Виявлено, що більшість існуючих аналогів характеризується перенасиченістю функціоналу та складним інтерфейсом.

У відповідності до функціональних вимог, які були вказані у технічному завданні, була створена компонентна та функціональна структури програмного застосунку. На наступному етапі проєктування було обрано мову програмування, середовище розробки та реалізовано необхідні програмні класи для забезпечення функціональності застосунку. Крім того, було спроєктовано та реалізовано графічний інтерфейс користувача.

Практична цінність розробленого програмного застосунку полягає в тому, що він надає користувачеві можливості перегляду змісту файлової системи та виконання множини операцій керування файловою системою: створення, видалення, копіювання, вставки, вирізання, перейменування, відкриття файлів, а також зручне налаштовування інтерфейсу програми.

Програмна документація включає: технічне завдання; керівництво програміста; керівництво оператора та тексти основних програмних модулів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Total Commander [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Total_Commander.
2. Призначення і функції програми Total Commander» [Електронний ресурс]. – Режим доступу: <http://um.co.ua/8/8-17/8-171757.html>.
3. FAR Manager [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/FAR_Manager.
4. Провідник Windows [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Провідник_Windows.
5. Мова програмування [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Мова_програмування.
6. С #: що це за мова та де її використовують [Електронний ресурс]. – Режим доступу: <https://robotdreams.cc/uk/blog/284-s-hto-eto-za-yazyk-i-gde-ego-ispolzuyut>.
7. С# [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp.
8. 9 причин вивчити мову С# [Електронний ресурс]. – Режим доступу: <https://foxminded.ua/9-prichin-vivchiti-movu-c/>.
9. Інтегроване середовище розробки [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Інтегроване_середовище_розробки.
10. Perkins, B. C# 6 Programming with Visual Studio 2015 / B. Perkins, Jon D. Reid, 2015. – 840 p.
11. Desktop Guide (Windows Forms .NET) [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-6.0>.
12. Welcome to the Visual Studio IDE [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022>.

13. Вимоги до системи для Windows 10 [Електронний ресурс]. – Режим доступу: <https://support.microsoft.com/uk-ua/windows/вимоги-до-системи--для-windows-10-6d4e9a79-66bf-7950-467c-795cf0386715>.

ДОДАТОК А
Текст програми

A.1 Код файлу MainForm

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Text;
using System.Windows.Forms;
using Aspose.Cells;

namespace FreeFile
{
    public partial class MainForm : System.Windows.Forms.Form
    {
        // Завантаження форми //
        public MainForm()
        {
            InitializeComponent();
            Directory.Add(@"C:\");
            UpdateDisk();
            UpdateDir();
            SidebarList.ContextMenuStrip = DiskContextMenu;
            FileList.ContextMenuStrip = FileContextMenuStrip;
            ToolTipLoad();// завантажуюмо спливаючі підказки
            SetStyle();
            SetLanguage();
        }
        // Глобальні змінні //
        List<string> Directory = new List<string>(); //
        Поточна директорія
        List<FileObject> CurrentFiles = new List<FileObject>(); //
        Зміст поточної директорії
        List<DiskObject> CurrentDisks = new List<DiskObject>(); //
        Підключені диски
        List<ClipboardObject> Clipboard = new List<ClipboardObject>(); //
        Буфер обміну
        bool Cut = false; // Прапорець для вирізання
        string newName = null; // Змінна для перейменування
        public string NewName { get => newName; set => newName = value; }

        string name402 = null; // Змінна для перейменування при однакових
        іменах
        public string Name402 { get => name402; set => name402 = value; }
        string FormStyle = "Standart";
        string FormLanguage = "Ukrainian";
        // Робота з дирикторією //
        bool LoadDir(List<string> Dir)
        {
            try
            {
                // Формуємо строку шляху для зміни директорії в консолі
                string Path = Dir[0];
                if (Dir.Count == 1) Path += @"\";
                for (int i = 1; i < Dir.Count; i++)
                {
                    Path += @"\ " + Dir[i];
                }
                string BatPath = "dir.bat";
            }
            catch { }
        }
    }
}

```

```

        string Text = @"chcp 65001 > nul" + '\n' + "cd /d " + Path +
'\n' + "chcp 1251 > nul" + '\n' + "dir > %~dp0dir.txt";
        // chcp 65001 > nul для зчитування юнікоду
        // cd /d " + Path змінює директорию в консолі
        // chcp 1251 > nul для зчитування кирилиці
        // dir > %~dp0dir.txt записуємо зміст відкритої директорії у
файл dir.txt
        // повний перезапис бат файлу
        using (StreamWriter writer = new StreamWriter(BatPath, false))
        {
            writer.WriteLine(Text);
        }
        // запускаємо бат файл
        ProcessStartInfo psi = new ProcessStartInfo();
        // Ім'я запускового приложення
        psi.FileName = "cmd";
        // команда, которую надо выполнить
        psi.Arguments = @"/c " + BatPath;
        // /c - после выполнения команды консоль закрывается
        // /k - не закрывать консоль после выполнения команды
        Process.Start(psi);
        System.Threading.Thread.Sleep(100);
        return true;
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 503;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        return false;
    }
}

bool ReadDir(string Path) // Зчитування файлу зі змістом поточної
директорії
{
    StreamReader sr = new StreamReader(Path,
Encoding.GetEncoding(1251));
    try
    {
        CurrentFiles.Clear();
        string line;
        while (!sr.EndOfStream)
        {
            line = sr.ReadLine();
            if (line != "")
            {
                if (line[0].ToString() != " ")
                    if (line[36] != '.')
                    {
                        FileObject CurrentFile = new FileObject();
                        int i = 0;
                        DataAndTime DaT = new DataAndTime();
                        while (line[i] != ' ')
                        {
                            DaT.Data += line[i];
                            i++;
                        }
                        i += 2;
                        while (line[i] != ' ')

```

```

{
    DaT.Time += line[i];
    i++;
}
CurrentFile.ChangeDate = DaT;
while (line[i] == ' ')
{
    i++;
}
string Size = "";
if ((int)line[i] > 48 && (int)line[i] < 57)
{
    while (i != 36)
    {
        Size += line[i];
        i++;
    }
}
else
{
    while (line[i] != ' ')
    {
        Size += line[i];
        i++;
    }
    while (line[i] == ' ')
    {
        i++;
    }
}
if (Size == "<DIR>") Size = "";
else Size += " K6";
CurrentFile.Size = Size;
string Name = "";
string Type = "";
if (Size == "")
{
    while (i < line.Length)
    {
        Name += line[i];
        i++;
    }
    Type = "Folder";
}
else
{
    while (line[i] != '.')
    {
        Name += line[i];
        i++;
    }
    while (i < line.Length)
    {
        Type += line[i];
        i++;
    }
}
CurrentFile.Name = Name;
CurrentFile.Type = Type;
CurrentFiles.Add(CurrentFile);
}

```

```

    }
    }
    sr.Close();
    return true;
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 503;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
    sr.Close();
    return false;
}
}
bool UpdateDir() // Оновлення змісту поточної директорії
{
    UpdateDisk();
    if (LoadDir(Dirictory))
        if (ReadDir("dir.txt"))
            if (UpdateFileList(CurrentFiles))
                return true;
    return false;
}
// Робота з дисками //
bool LoadDisk() // Завантаження до файлу списку підключених дисків
{
    try
    {
        ProcessStartInfo psi = new ProcessStartInfo();
        psi.FileName = "cmd";
        psi.Arguments = @"/c wmic logicaldisk get
size,freespace,caption > disk.txt";
        Process.Start(psi);
        System.Threading.Thread.Sleep(300);
        return true;
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 503;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        return false;
    }
}
bool ReadDisk(string Path) // Зчитування файлу зі списком підключених
дисків
{
    StreamReader sr = new StreamReader(Path,
Encoding.GetEncoding(1251));
    try
    {
        CurrentDisks.Clear();
        string line;
        line = sr.ReadLine();
        while (!sr.EndOfStream)
        {
            line = sr.ReadLine();

```

```

        DiskObject NewDisk = new DiskObject();
        int i = 0;
        while (line[i] != ' ')
        {
            NewDisk.Caption += line[i];
            i++;
        }
        NewDisk.Caption += @"\\";
        while (line[i] == ' ')
            i++;
        while (line[i] != ' ')
        {
            NewDisk.FreeSpace += line[i];
            i++;
        }
        while (line[i] == ' ')
            i++;
        while (line[i] != ' ')
        {
            NewDisk.Size += line[i];
            i++;
        }
        CurrentDisks.Add(NewDisk);
    }
    sr.Close();
    return true;
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 503;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
    sr.Close();
    return false;
}
}
bool UpdateDisk() // Оновлення підключених дисків
{
    if(LoadDisk())
        if(ReadDisk("disk.txt"))
            if(UpdateSidebarList(CurrentDisks))
                return true;
    return false;
}
// Оновлення списків //
bool UpdateFileList(List<FileObject> Files)
{
    try
    {
        FileList.Items.Clear();
        ImageList imageList = new ImageList();
        imageList.ImageSize = new Size(16, 16);
        imageList.Images.Add(new Bitmap("../images/acses.png"));
//0 accdb
        imageList.Images.Add(new Bitmap("../images/audio.png"));
//1 mp3 ogg
        imageList.Images.Add(new Bitmap("../images/arxiv.png"));
//2 rar zip
    }
    catch { }
}

```

```

        imageList.Images.Add(new Bitmap("./images/bloknot.png"));
//3 txt
        imageList.Images.Add(new Bitmap("./images/excel.png"));
//4 xlsx
        imageList.Images.Add(new Bitmap("./images/exe.png"));
//5 exe
        imageList.Images.Add(new Bitmap("./images/folder.png"));
//6 Folder
        imageList.Images.Add(new Bitmap("./images/file.png"));
//7 незаписане
        imageList.Images.Add(new Bitmap("./images/foto.png"));
//8 png bmp
        imageList.Images.Add(new Bitmap("./images/pdf.png"));
//9 pdf
        imageList.Images.Add(new Bitmap("./images/video.png"));
//10 mp4 avi mov
        imageList.Images.Add(new Bitmap("./images/word.png"));
//11 docx doc
        imageList.Images.Add(new
Bitmap("./images/presentation.png")); //12 pptx
        FileList.SmallImageList = imageList;
        for (int i = 0; i < Files.Count; i++)
        {
            string Type = Files[i].Type;
            if (Files[i].Type == "Folder")
            {
                switch (FormLanguage)
                {
                    case "Ukrainian":
                        Type = "Папка";
                        break;
                    case "English":
                        break;
                    default:
                        break;
                }
            }
            ListViewItem item = new ListViewItem(new string[] {
Files[i].Name,
Files[i].ChangeDate.Data + " " +
Files[i].ChangeDate.Time,
Type, Files[i].Size });
            if (Files[i].Type == "Folder")
            {
                item.ImageIndex = 6;
            }
            else
            if (Files[i].Type == ".txt")
            {
                item.ImageIndex = 3;
            }
            else
            if ((Files[i].Type == ".docx") || (Files[i].Type ==
".doc"))
            {
                item.ImageIndex = 11;
            }
            else
            if (Files[i].Type == ".accdb")
            {
                item.ImageIndex = 0;
            }
        }
    }
}

```

```

    }
    else
    if (Files[i].Type == ".xlsx")
    {
        item.ImageIndex = 4;
    }
    else
    if ((Files[i].Type == ".rar") || (Files[i].Type ==
".zip"))
    {
        item.ImageIndex = 2;
    }
    else
    if ((Files[i].Type == ".png") || (Files[i].Type ==
".bmp"))
    {
        item.ImageIndex = 8;
    }
    else
    if ((Files[i].Type == ".mp3") || (Files[i].Type ==
".ogg"))
    {
        item.ImageIndex = 1;
    }
    else
    if ((Files[i].Type == ".mp4") || (Files[i].Type == ".avi")
|| (Files[i].Type == ".mov"))
    {
        item.ImageIndex = 10;
    }
    else
    if (Files[i].Type == ".exe")
    {
        item.ImageIndex = 5;
    }
    else
    if (Files[i].Type == ".pdf")
    {
        item.ImageIndex = 9;
    }
    else
    if (Files[i].Type == ".pptx")
    {
        item.ImageIndex = 12;
    }
    else
        item.ImageIndex = 7;
    FileList.Items.Add(item);
}
UpdateDirictoryList(Dirictory);
return true;
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 503;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
    return false;
}

```

```

}
bool UpdateDirictoryList(List<string> Dir)
{
    try
    {
        DirictoryList.Items.Clear();
        for (int i = 0; i < Dir.Count; i++)
        {
            DirictoryList.Items.Add(Dir[i]);
        }
        return true;
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 503;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        return false;
    }
}

bool UpdateSidebarList(List<DiskObject> Disks)
{
    try
    {
        SidebarList.Items.Clear();
        ImageList imageList = new ImageList();
        imageList.ImageSize = new Size(16, 16);
        imageList.Images.Add(new Bitmap("./images/disk.png"));
        SidebarList.SmallImageList = imageList;
        for (int i = 0; i < Disks.Count; i++)
        {
            ListViewItem item = new ListViewItem(new string[] {
Disks[i].Caption, "test" });
            item.ImageIndex = 0;
            SidebarList.Items.Add(item);
        }
        return true;
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 503;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        return false;
    }
}

// Кліки на списки //
private void FileList_DoubleClick(object sender, EventArgs e)
{
    if (FileList.SelectedItems.Count == 1)
    {
        if ((FileList.SelectedItems[0].SubItems[2].Text != "Folder")
&& (FileList.SelectedItems[0].SubItems[2].Text != "Папка"))
        {
            OpenFile(Dirictory, FileList.SelectedItems[0].Text +
FileList.SelectedItems[0].SubItems[2].Text);
        }
    }
}

```

```

        else
        {
            Dirictory.Add(FileList.SelectedItems[0].Text);
            if (UpdateDir() == false)
            {
                Dirictory.Remove(FileList.SelectedItems[0].Text);
                UpdateDir();
            }
        }
    }
}
private void DirictoryList_Click(object sender, EventArgs e)
{
    if (DirictoryList.SelectedItems.Count == 1)
    {
        int Index =
DirictoryList.Items.IndexOf(DirictoryList.SelectedItems[0]);
        for (int i = Dirictory.Count - 1; i > Index; i--)
        {
            Dirictory.RemoveAt(i);
        }
        UpdateDir();
    }
}
private void SidebarList_Click(object sender, MouseEventArgs e) //
Відкриття з меню диска
{
    if ((SidebarList.SelectedItems.Count == 1) && (e.Button ==
MouseButtons.Left))
    {
        Dirictory.Clear();
        Dirictory.Add(SidebarList.SelectedItems[0].Text);
        UpdateDir();
    }
}
private void SidebarList_Click(object sender, EventArgs e) //
Відкриття лівою кнопкою миші
{
    if (SidebarList.SelectedItems.Count == 1)
    {
        Dirictory.Clear();
        Dirictory.Add(SidebarList.SelectedItems[0].Text);
        UpdateDir();
    }
}
// Операції з фалами //
bool OpenFile(List<string> Dir, string FileName)
{
    try
    {
        Process.Start(BuildPath(Dir, FileName));
        return true;
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 504;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        return false;
    }
}

```

```

    }
}
private void CopyFile(object sender, EventArgs e)
{
    try
    {
        Cut = false;
        if ((FileList.SelectedItems[0].SubItems[2].Text != "Folder")
        && (FileList.SelectedItems[0].SubItems[2].Text != "Папка"))
        {
            Clipboard.Clear();
            ClipboardObject temp = new ClipboardObject();
            temp.Path = BuildPath(Dirictory,
FileList.SelectedItems[0].Text + FileList.SelectedItems[0].SubItems[2].Text);
            temp.Type = "File";
            temp.Name = FileList.SelectedItems[0].Text +
FileList.SelectedItems[0].SubItems[2].Text;
            Clipboard.Add(temp);
        }
        else
        {
            Clipboard.Clear();
            ClipboardObject temp = new ClipboardObject();
            temp.Path = BuildPath(Dirictory,
FileList.SelectedItems[0].Text);
            temp.Type = "Folder";
            temp.Name = FileList.SelectedItems[0].Text;
            Clipboard.Add(temp);
        }
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 505;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
    }
}
private void InsertFile(object sender, EventArgs e)
{
    try
    {
        for (int i = 0; i < Clipboard.Count; i++)
        {
            if ((Clipboard[i].Type == "Folder") || (Clipboard[i].Type
== "Папка"))
            {
                CopyDir(Clipboard[i].Path, BuildPath(Dirictory) + "/"
+ Clipboard[i].Name);
                if (Cut)
                {
                    Directory.Delete(Clipboard[i].Path, true);
                    Cut = false;
                }
            }
            else
            {
                File.Copy(Clipboard[i].Path, BuildPath(Dirictory,
Clipboard[i].Name), true);
                if (Cut)

```

```

        {
            File.Delete(Clipboard[i].Path);
            Cut = false;
        }
    }
    Clipboard.Clear();
    UpdateDir();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
private void DeleteFile(object sender, EventArgs e)
{
    try
    {
        if ((FileList.SelectedItems[0].SubItems[2].Text != "Folder")
        && (FileList.SelectedItems[0].SubItems[2].Text != "Папка"))
        {
            File.Delete(BuildPath(Dirictory,
            FileList.SelectedItems[0].Text
            FileList.SelectedItems[0].SubItems[2].Text));
        }
        else
        {
            Directory.Delete(BuildPath(Dirictory,
            FileList.SelectedItems[0].Text), true);
        }
        UpdateDir();
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 501;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
    }
}
private void CreateTextFile(object sender, EventArgs e)
{
    bool exist = false;
    try
    {
        switch (FormLanguage)
        {
            case "Ukrainian":
                for (int i = 0; i < CurrentFiles.Count; i++)
                    if ((CurrentFiles[i].Type == ".txt") &&
                    (CurrentFiles[i].Name == "НовийФайл"))
                        exist = true;
                if (exist == false)
                    File.Create(BuildPath(Dirictory,
                    "НовийФайл.txt")).Close();
                break;

```

```

        case "English":
            for (int i = 0; i < CurrentFiles.Count; i++)
                if ((CurrentFiles[i].Type == ".txt") &&
(CurrentFiles[i].Name == "NewFile"))
                    exist = true;
            if (exist == false)
                File.Create(BuildPath(Dirictory,
"NewFile.txt")).Close();
            break;
        default:
            break;
    }
    if (exist)
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 402;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        switch (FormLanguage)
        {
            case "Ukrainian":
                Create402("НовийФайл", ".txt");
                break;
            case "English":
                Create402("NewFile", ".txt");
                break;
            default:
                break;
        }
    }
    else
        UpdateDir();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
private void CreateFolder(object sender, EventArgs e)
{
    bool exist = false;
    try
    {
        switch (FormLanguage)
        {
            case "Ukrainian":
                for (int i = 0; i < CurrentFiles.Count; i++)
                    if (((CurrentFiles[i].Type == "Folder") ||
(CurrentFiles[i].Type == "Папка")) && (CurrentFiles[i].Name == "НоваПапка"))
                        exist = true;
                if (exist == false)
                    Directory.CreateDirectory(BuildPath(Dirictory,
"НоваПапка"));
                break;
            case "English":
                for (int i = 0; i < CurrentFiles.Count; i++)

```

```

        if (((CurrentFiles[i].Type == "Folder") ||
(CurrentFiles[i].Type == "Папка")) && (CurrentFiles[i].Name ==
"NewDirectory"))
            exist = true;
        if (exist == false)
            Directory.CreateDirectory(BuildPath(Dirictory,
"NewDirectory"));
        break;
    default:
        break;
    }
    if (exist)
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 402;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        switch (FormLanguage)
        {
            case "Ukrainian":
                Create402("НоваПапка", "");
                break;
            case "English":
                Create402("NewDirectory", "");
                break;
            default:
                break;
        }
    }
    else
        UpdateDir();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
private void CreateBmpFile(object sender, EventArgs e)
{
    bool exist = false;
    try
    {
        switch (FormLanguage)
        {
            case "Ukrainian":
                for (int i = 0; i < CurrentFiles.Count; i++)
                    if ((CurrentFiles[i].Type == ".bmp") &&
(CurrentFiles[i].Name == "НовийМалюнок"))
                        exist = true;
                if (exist == false)
                {
                    Bitmap bitmap = new Bitmap(100, 100);
                    bitmap.Save(BuildPath(Dirictory,
"НовийМалюнок.bmp"));
                }
                break;

```

```

        case "English":
            for (int i = 0; i < CurrentFiles.Count; i++)
                if ((CurrentFiles[i].Type == ".bmp") &&
(CurrentFiles[i].Name == "NewPicture"))
                    exist = true;
            if (exist == false)
            {
                Bitmap bitmap = new Bitmap(100, 100);
                bitmap.Save(BuildPath(Dirictory,
"NewPicture.bmp"));
            }
            break;
        default:
            break;
    }
    if (exist)
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 402;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        switch (FormLanguage)
        {
            case "Ukrainian":
                Create402("НовийМалюнок", ".bmp");
                break;
            case "English":
                Create402("NewPicture", ".bmp");
                break;
            default:
                break;
        }
    }
    else
        UpdateDir();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
private void CreateDocxFile(object sender, EventArgs e)
{
    bool exist = false;
    try
    {
        switch (FormLanguage)
        {
            case "Ukrainian":
                for (int i = 0; i < CurrentFiles.Count; i++)
                    if ((CurrentFiles[i].Type == ".docx") &&
(CurrentFiles[i].Name == "НовийДокумент"))
                        exist = true;
                if (exist == false)
                    File.Create(BuildPath(Dirictory,
"НовийДокумент.docx")).Close();

```

```

        break;
    case "English":
        for (int i = 0; i < CurrentFiles.Count; i++)
            if ((CurrentFiles[i].Type == ".docx") &&
(CurrentFiles[i].Name == "NewDocument"))
                exist = true;
            if (exist == false)
                File.Create(BuildPath(Dirictory,
"NewDocument.docx")).Close();
                break;
            default:
                break;
        }
    if (exist)
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 402;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();

        switch (FormLanguage)
        {
            case "Ukrainian":
                Create402("НовийДокумент", ".docx");
                break;

            case "English":
                Create402("NewDocument", ".docx");
                break;

            default:
                break;
        }
    }
    else
        UpdateDir();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
private void CreateXlsxFile(object sender, EventArgs e)
{
    bool exist = false;
    try
    {
        switch (FormLanguage)
        {
            case "Ukrainian":
                for (int i = 0; i < CurrentFiles.Count; i++)
                    if ((CurrentFiles[i].Type == ".xlsx") &&
(CurrentFiles[i].Name == "НоваТаблиця"))
                        exist = true;
                    if (exist == false)
                    {

```

```

        Workbook wb = new Workbook();
        wb.Save(BuildPath(Dirictory, "НоваТаблиця.xlsx"),
SaveFormat.Xlsx);
    }
    break;
    case "English":
        for (int i = 0; i < CurrentFiles.Count; i++)
            if ((CurrentFiles[i].Type == ".xlsx") &&
(CurrentFiles[i].Name == "NewSpreadsheet"))
                exist = true;
            if (exist == false)
            {
                Workbook wb = new Workbook();
                wb.Save(BuildPath(Dirictory,
"NewSpreadsheet.xlsx"), SaveFormat.Xlsx);
            }
            break;
        default:
            break;
    }
    if (exist)
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 402;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        switch (FormLanguage)
        {
            case "Ukrainian":
                Create402("НоваТаблиця", ".xlsx");
                break;
            case "English":
                Create402("NewSpreadsheet", ".xlsx");
                break;
            default:
                break;
        }
    }
    else
        UpdateDir();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
private void CreatePptxFile(object sender, EventArgs e)
{
    bool exist = false;
    try
    {
        switch (FormLanguage)
        {
            case "Ukrainian":
                for (int i = 0; i < CurrentFiles.Count; i++)

```

```

        if ((CurrentFiles[i].Type == ".pptx") &&
(CurrentFiles[i].Name == "НоваПрезентація"))
            exist = true;
        if (exist == false)
            File.Create(BuildPath(Dirictory,
"НоваПрезентація.pptx")).Close();
        break;
        case "English":
            for (int i = 0; i < CurrentFiles.Count; i++)
                if ((CurrentFiles[i].Type == ".pptx") &&
(CurrentFiles[i].Name == "NewPresentation"))
                    exist = true;
            if (exist == false)
                File.Create(BuildPath(Dirictory,
"NewPresentation.pptx")).Close();
            break;
        default:
            break;
    }
    if (exist)
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 402;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        switch (FormLanguage)
        {
            case "Ukrainian":
                Create402("НоваПрезентація", ".pptx");
                break;
            case "English":
                Create402("NewPresentation", ".pptx");
                break;
            default:
                break;
        }
    }
    else
        UpdateDir();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
private void CutFile(object sender, EventArgs e)
{
    try
    {
        CopyFile(sender, e);
        Cut = true;
        UpdateDir();
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();

```

```

        errorForm.Code = 505;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
    }
}
private void RenameFile(object sender, EventArgs e)
{
    try
    {
        RenamingForm RForm = new RenamingForm();
        RForm.Main = this;
        RForm.FormStyle = FormStyle;
        RForm.FormLanguage = FormLanguage;
        RForm.OldName = FileList.SelectedItems[0].Text;
        RForm.ShowDialog();
        if (NewName != null)
        {
            if ((FileList.SelectedItems[0].SubItems[2].Text !=
"Folder") && (FileList.SelectedItems[0].SubItems[2].Text != "Папка"))
            {
                File.Move(BuildPath(Dirictory,
FileList.SelectedItems[0].Text + FileList.SelectedItems[0].SubItems[2].Text),
BuildPath(Dirictory, NewName +
FileList.SelectedItems[0].SubItems[2].Text));
            }
            else
            {
                Directory.Move(BuildPath(Dirictory,
FileList.SelectedItems[0].Text),
BuildPath(Dirictory, NewName));
            }
            UpdateDir();
        }
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 402;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
    }
}
private void PropertiesFile(object sender, EventArgs e)
{
    try
    {
        FilePropertiesForm filePropertiesForm = new
FilePropertiesForm();
        if ((FileList.SelectedItems[0].SubItems[2].Text != "Folder")
&& (FileList.SelectedItems[0].SubItems[2].Text != "Папка"))
        {
            filePropertiesForm.Path = BuildPath(Dirictory,
FileList.SelectedItems[0].Text + FileList.SelectedItems[0].SubItems[2].Text);
            filePropertiesForm.Type = "File";
        }
        else
        {
            filePropertiesForm.Path = BuildPath(Dirictory,
FileList.SelectedItems[0].Text);

```

```

        filePropertiesForm.Type = "Folder";
    }
    filePropertiesForm.FormStyle = FormStyle;
    filePropertiesForm.FormLanguage = FormLanguage;
    filePropertiesForm.Show();
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
}
}
// Допоміжні функції //
string BuildPath(List<string> Dir, string FileName) // для файлу
{
    try
    {
        string Path = Dir[0];
        for (int i = 1; i < Dir.Count; i++)
        {
            if (i != 1)
                Path += @"\";
            Path += Dir[i];
        }
        if (Dir.Count != 1)
            Path += @"\";
        Path += FileName;
        return Path;
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 502;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
        return null;
    }
}
string BuildPath(List<string> Dir) // для папки
{
    try
    {
        string Path = Dir[0];
        for (int i = 1; i < Dir.Count; i++)
        {
            if (i != 1)
                Path += @"\";
            Path += Dir[i];
        }
        return Path;
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 502;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;

```

```

        errorForm.ShowDialog();
        return null;
    }
}
void CopyDir(string FromDir, string ToDir) // рекурсивна функція для
копіювання змісту папки
{
    try
    {
        Directory.CreateDirectory(ToDir);
        foreach (string s1 in Directory.GetFiles(FromDir))
        {
            string s2 = ToDir + "\\\" + Path.GetFileName(s1);
            File.Copy(s1, s2);
        }
        foreach (string s in Directory.GetDirectories(FromDir))
        {
            CopyDir(s, ToDir + "\\\" + Path.GetFileName(s));
        }
    }
    catch
    {
        ErrorForm errorForm = new ErrorForm();
        errorForm.Code = 505;
        errorForm.FormStyle = FormStyle;
        errorForm.FormLanguage = FormLanguage;
        errorForm.ShowDialog();
    }
}
bool Create402(string OldName, string Type)// Створення файлу при
помилці 402
{
    try
    {
        bool exist = false;
        RenamingForm RForm = new RenamingForm();
        RForm.Main = this;
        RForm.FormStyle = FormStyle;
        RForm.FormLanguage = FormLanguage;
        RForm.OldName = OldName;
        RForm.Error402 = true;
        RForm.ShowDialog();
        if (Name402 != null)
            if (Type == "")
            {
                for (int i = 0; i < CurrentFiles.Count; i++)
                    if (((CurrentFiles[i].Type == "Folder") ||
(CurrentFiles[i].Type == "Папка")) && (CurrentFiles[i].Name == Name402))
                        exist = true;
                if (exist == false)
                    Directory.CreateDirectory(BuildPath(Dirictory,
Name402));
            }
            else
            {
                for (int i = 0; i < CurrentFiles.Count; i++)
                    if ((CurrentFiles[i].Type == Type) &&
(CurrentFiles[i].Name == Name402))
                        exist = true;
                if (exist == false)
                {

```

```

        if (Type == ".bmp")
        {
            Bitmap bitmap = new Bitmap(100, 100);
            bitmap.Save(BuildPath(Dirictory, Name402 +
Type));
        }
        else if (Type == ".xlsx")
        {
            Workbook wb = new Workbook();
            wb.Save(BuildPath(Dirictory, Name402 + Type),
SaveFormat.Xlsx);
        }
        else
            File.Create(BuildPath(Dirictory, Name402 +
Type)).Close();
    }
}
if (exist)
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 402;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
    Create402(Name402, Type);
}
else
    UpdateDir();
return true;
}
catch
{
    ErrorForm errorForm = new ErrorForm();
    errorForm.Code = 505;
    errorForm.FormStyle = FormStyle;
    errorForm.FormLanguage = FormLanguage;
    errorForm.ShowDialog();
    return false;
}
}
// Випадаючі меню //
private void DiskContextMenu_Opening(object sender, CancelEventArgs e)
// випадаюче меню для дисків
{
    if (SidebarList.SelectedItems.Count != 1)
    {
        e.Cancel = true;
    }
}
private void DiskInfo_Click(object sender, EventArgs e) // відкриття
форми властивостей диску
{
    if (SidebarList.SelectedItems.Count == 1)
    {
        DiskObject SelectedDisk = new DiskObject();
        for (int i = 0; i < CurrentDisks.Count; i++)
            if (CurrentDisks[i].Caption ==
SidebarList.SelectedItems[0].Text)
                SelectedDisk = CurrentDisks[i];
        DiskInfoForm NewForm = new DiskInfoForm(SelectedDisk);
        NewForm.FormStyle = FormStyle;
    }
}

```

```

        NewForm.FormLanguage = FormLanguage;
        NewForm.Show();
    }
}
private void FileContextMenuStrip_Opening(object sender,
CancelEventArgs e) // випадаюче меню для файлів та папок
{
    if (FileList.SelectedItems.Count > 0)
    {
        FileContextMenuStrip.Items[0].Visible = true;
        FileContextMenuStrip.Items[1].Visible = true;
        FileContextMenuStrip.Items[2].Visible = true;
        FileContextMenuStrip.Items[3].Visible = true;
        FileContextMenuStrip.Items[4].Visible = true;
        FileContextMenuStrip.Items[5].Visible = true;
        FileContextMenuStrip.Items[6].Visible = false;
        FileContextMenuStrip.Items[7].Visible = false;
    }
    else
    {
        FileContextMenuStrip.Items[0].Visible = false;
        FileContextMenuStrip.Items[1].Visible = false;
        FileContextMenuStrip.Items[2].Visible = false;
        FileContextMenuStrip.Items[3].Visible = false;
        FileContextMenuStrip.Items[4].Visible = false;
        FileContextMenuStrip.Items[5].Visible = false;
        FileContextMenuStrip.Items[6].Visible = true;
        FileContextMenuStrip.Items[7].Visible = true;
    }
}
private void HelpToolStripMenuItem_Click(object sender, EventArgs e)
// відкриття форми справка
{
    HelpForm helpForm = new HelpForm();
    helpForm.FormStyle = FormStyle;
    helpForm.FormLanguage = FormLanguage;
    helpForm.Show();
}
// Підказки //
private void ToolTipLoad()
{
    ToolTip.SetToolTip(DirictoryList, "Поточна директорія");
    ToolTip.SetToolTip(FileList, "Зміст поточної директорії");
    ToolTip.SetToolTip(SidebarList, "Підключені диски");
}
private void About(object sender, EventArgs e) // відкриття форми про
програму
{
    AboutForm aboutForm = new AboutForm();
    aboutForm.FormStyle = FormStyle;
    aboutForm.FormLanguage = FormLanguage;
    aboutForm.Show();
}
// Теми
private void StandartRadioButton_CheckedChanged(object sender,
EventArgs e)
{
    if (StandartRadioButton.Checked)
    {
        FormStyle = "Standart";
        SetStyle();
    }
}

```

```

    }
}
private void BlackRadioButton_CheckedChanged(object sender, EventArgs
e)
{
    if (DarkRadioButton.Checked)
    {
        FormStyle = "Black";
        SetStyle();
    }
}
private void LightRadioButton_CheckedChanged(object sender, EventArgs
e)
{
    if (LightRadioButton.Checked)
    {
        FormStyle = "Light";
        SetStyle();
    }
}
// Мови
private void UkrainianRadioButton_CheckedChanged(object sender,
EventArgs e)
{
    if (UkrainianRadioButton.Checked)
    {
        FormLanguage = "Ukrainian";
        SetLanguage();
        UpdateFileList(CurrentFiles);
    }
}
private void EnglishRadioButton_CheckedChanged(object sender,
EventArgs e)
{
    if (EnglishRadioButton.Checked)
    {
        FormLanguage = "English";
        SetLanguage();
        UpdateFileList(CurrentFiles);
    }
}
// Функції налаштування форми //
void SetStyle()
{
    Style StyleSeter = new Style();
    StyleSeter.SetStyleSetting(FormStyle);
    this.BackColor = StyleSeter.BackColor;
    FileList.OwnerDraw = true;
    FileList.DrawColumnHeader += new
DrawListViewColumnHeaderEventHandler(List_DrawColumnHeader);
    FileList.DrawItem += new
DrawListViewItemEventHandler(List_DrawItem);
    FileList.DrawSubItem += new
DrawListViewSubItemEventHandler(List_DrawSubItem);
    FileList.ForeColor = StyleSeter.ForeColor;
    FileList.BackColor = StyleSeter.ThirdColor;

    DirictoryList.OwnerDraw = true;
    DirictoryList.DrawColumnHeader += new
DrawListViewColumnHeaderEventHandler(List_DrawColumnHeader);

```

```

        DirectoryList.DrawItem += new
DrawListViewItemEventHandler(DirList_DrawItem);
        DirectoryList.DrawSubItem += new
DrawListViewSubItemEventHandler(List_DrawSubItem);
        DirectoryList.ForeColor = StyleSetter.ForeColor;
        DirectoryList.BackColor = StyleSetter.ThirdColor;
        SidebarList.OwnerDraw = true;
        SidebarList.DrawColumnHeader += new
DrawListViewColumnHeaderEventHandler(List_DrawColumnHeader);
        SidebarList.DrawItem += new
DrawListViewItemEventHandler(List_DrawItem);
        SidebarList.DrawSubItem += new
DrawListViewSubItemEventHandler(List_DrawSubItem);
        SidebarList.ForeColor = StyleSetter.ForeColor;
        SidebarList.BackColor = StyleSetter.ThirdColor;
        void List_DrawColumnHeader(object sender,
DrawListViewColumnHeaderEventArgs e)
        {
            e.Graphics.FillRectangle(new
SolidBrush(StyleSetter.ThirdColor), e.Bounds);
            e.Graphics.DrawString(e.Header.Text, e.Font, new
SolidBrush(StyleSetter.ForeColor), e.Bounds);
        }
        void List_DrawItem(object sender, DrawListViewItemEventArgs e)
        {
            if ((e.State & ListViewItemStates.Selected) != 0)
            {
                // Draw the background and focus rectangle for a selected
item.
                e.Graphics.FillRectangle(new
SolidBrush(StyleSetter.ThirdColor), e.Bounds);
            }
            else
            {
                // Draw the background for an unselected item.
                e.Graphics.FillRectangle(new
SolidBrush(StyleSetter.ThirdColor), e.Bounds);
            }
        }
        void DirList_DrawItem(object sender, DrawListViewItemEventArgs e)
        {
            if ((e.State & ListViewItemStates.Selected) != 0)
            {
                // Draw the background and focus rectangle for a selected
item.
                e.Graphics.FillRectangle(new
SolidBrush(StyleSetter.ThirdColor), e.Bounds);
            }
            else
            {
                // Draw the background for an unselected item.
                e.Graphics.FillRectangle(new
SolidBrush(StyleSetter.ThirdColor), e.Bounds);
            }
            e.DrawText();
        }
        void List_DrawSubItem(object sender, DrawListViewSubItemEventArgs
e)
        {
            e.DrawDefault = true;
        }

```

```

MenuStrip.Renderer = StyleSetter.Renderer;
MenuStrip.BackColor = StyleSetter.ThirdColor;
MenuStrip.ForeColor = StyleSetter.ForeColor;
справкаToolStripMenuItem.BackColor = StyleSetter.ThirdColor;
справкаToolStripMenuItem.ForeColor = StyleSetter.ForeColor;
проПрограмуToolStripMenuItem.BackColor = StyleSetter.ThirdColor;
проПрограмуToolStripMenuItem.ForeColor = StyleSetter.ForeColor;
// Кнопки
CopyFileButton.FillColor = StyleSetter.MainColor;
CopyFileButton.FillColor2 = StyleSetter.SubColor;
CopyFileButton.BackColor = StyleSetter.BackColor;
CopyFileButton.BorderRadius = StyleSetter.CornerRadius;
CopyFileButton.ForeColor = StyleSetter.ForeColor;
InsertFileButton.FillColor = StyleSetter.MainColor;
InsertFileButton.FillColor2 = StyleSetter.SubColor;
InsertFileButton.BackColor = StyleSetter.BackColor;
InsertFileButton.BorderRadius = StyleSetter.CornerRadius;
InsertFileButton.ForeColor = StyleSetter.ForeColor;
OpenFileButton.FillColor = StyleSetter.MainColor;
OpenFileButton.FillColor2 = StyleSetter.SubColor;
OpenFileButton.BackColor = StyleSetter.BackColor;
OpenFileButton.BorderRadius = StyleSetter.CornerRadius;
OpenFileButton.ForeColor = StyleSetter.ForeColor;
RenameFileButton.FillColor = StyleSetter.MainColor;
RenameFileButton.FillColor2 = StyleSetter.SubColor;
RenameFileButton.BackColor = StyleSetter.BackColor;
RenameFileButton.BorderRadius = StyleSetter.CornerRadius;
RenameFileButton.ForeColor = StyleSetter.ForeColor;
// Меню
FileContextMenuStrip.Renderer = StyleSetter.Renderer;
FileContextMenuStrip.BackColor = StyleSetter.ThirdColor;
FileContextMenuStrip.ForeColor = StyleSetter.ForeColor;
текстовийФайлToolStripMenuItem.ForeColor = StyleSetter.ForeColor;
малюнокToolStripMenuItem.BackColor = StyleSetter.ThirdColor;
малюнокToolStripMenuItem.ForeColor = StyleSetter.ForeColor;
документWordToolStripMenuItem.BackColor = StyleSetter.ThirdColor;
документWordToolStripMenuItem.ForeColor = StyleSetter.ForeColor;
excelToolStripMenuItem.BackColor = StyleSetter.ThirdColor;
excelToolStripMenuItem.ForeColor = StyleSetter.ForeColor;
презентаціюToolStripMenuItem.BackColor = StyleSetter.ThirdColor;
презентаціюToolStripMenuItem.ForeColor = StyleSetter.ForeColor;
}
void SetLanguage()
{
    switch (FormLanguage)
    {
        case "Ukrainian":
            this.Text = "Файловий менеджер FreeFile";
            SidebarList.Columns[0].Text = "Диски";
            FileList.Columns[0].Text = "Назва";
            FileList.Columns[1].Text = "Дата зміни";
            FileList.Columns[2].Text = "Тип";
            FileList.Columns[3].Text = "Розмір";
            StyleGroupBox.Text = "Тема";
            LanguageGroupBox.Text = "Мова";
            StandartRadioButton.Text = "Стандартна";
            DarkRadioButton.Text = "Темна";
            LightRadioButton.Text = "Світла";
            UkrainianRadioButton.Text = "Українська";
            EnglishRadioButton.Text = "Англійська";
            CopyFileButton.Text = "Копіювати";

```

```

        InsertFileButton.Text = "Вставити";
        CutFileButton.Text = "Вирізати";
        OpenFileButton.Text = "Відкрити";
        PropertiesFileButton.Text = "Властивості";
        RenameFileButton.Text = "Перейменувати";
        файлToolStripMenuItem.Text = "Файл";
        справкаToolStripMenuItem.Text = "Справка";
        проПрограмуToolStripMenuItem.Text = "Про програму";
        відкритиToolStripMenuItem.Text = "Відкрити";
        властивостіToolStripMenuItem1.Text = "Властивості";
        break;
    case "English":
        this.Text = "File Manager FreeFile";
        SidebarList.Columns[0].Text = "Disks";
        FileList.Columns[0].Text = "Name";
        FileList.Columns[1].Text = "Date of change";
        FileList.Columns[2].Text = "Type";
        FileList.Columns[3].Text = "Size";
        StyleGroupBox.Text = "Theme";
        LanguageGroupBox.Text = "Language";
        StandartRadioButton.Text = "Standart";
        DarkRadioButton.Text = "Dark";
        LightRadioButton.Text = "Light";
        UkrainianRadioButton.Text = "Ukrainian";
        EnglishRadioButton.Text = "English";
        CopyFileButton.Text = "Copy";
        InsertFileButton.Text = "Insert";
        CutFileButton.Text = "Cut";
        OpenFileButton.Text = "Open";
        PropertiesFileButton.Text = "Properties";
        RenameFileButton.Text = "Rename";
        файлToolStripMenuItem.Text = "File";
        справкаToolStripMenuItem.Text = "Help";
        проПрограмуToolStripMenuItem.Text = "About the program";
        відкритиToolStripMenuItem.Text = "Open";
        властивостіToolStripMenuItem1.Text = "Properties";
        відкритиToolStripMenuItem1.Text = "Open";
        скопіюватиToolStripMenuItem.Text = "Copy";
        вирізатиToolStripMenuItem.Text = "Cut";
        перейменуватиToolStripMenuItem.Text = "Rename";
        видалитиToolStripMenuItem.Text = "Delete";
        властивостіToolStripMenuItem.Text = "Properties";
        створитиToolStripMenuItem.Text = "Create";
        вставитиToolStripMenuItem.Text = "Insert";
        папкуToolStripMenuItem.Text = "Folder";
        текстовийФайлToolStripMenuItem.Text = "Text file";
        малюнокToolStripMenuItem.Text = "Picture";
        документWordToolStripMenuItem.Text = "Word Document";
        excelToolStripMenuItem.Text = "Excel spreadsheet";
        презентаціюToolStripMenuItem.Text = "Presentation";
        break;
    default:
        break;
}
}
}
}
}

```

ДОДАТОК Б
Слайди презентації

ДИПЛОМНА РОБОТА

РОЗРОБКА СИСТЕМНОЇ УТИЛІТИ «ФАЙЛОВИЙ МЕНЕДЖЕР»

Розробив: ст. групи КНТ-119
Керівник: доцент, к.т.н.

Шевченко Я.О.
Сердюк С.М.

Рисунок Б.1 – Слайд №1

Мета роботи

розробка програми, що забезпечує користувачу взаємодію з файловою системою ОС Windows

Об'єкт дослідження – файлова система операційної системи (ОС) Windows

Предмет дослідження – методи взаємодії користувача з файловою системою



Задачі, що вирішуються

- провести аналіз предметної області;
- провести огляд програмних аналогів;
- розробка технічного завдання;
- розробка структури програмного забезпечення;
- розробка структури класів програмного забезпечення;
- програмна реалізація застосунку;
- розробка необхідної документації.



НУ "ЗП", Кафедра програмних засобів

Рисунок Б.3 – Слайд №3

3

Аналіз програм-аналогів

Характеристика	Total Commander	Far Manager	Провідник Windows
Можливість проводити операції з файлами	+	+	+
Підтримка плагінів	+	+	-
Безкоштовність	-	+	+
Дружній дизайн інтерфейсу	-	-	+
Можливість зручно налаштувати інтерфейс	+	-	-
Двопанельний інтерфейс	+	+	-



НУ "ЗП", Кафедра програмних засобів

Рисунок Б.4 – Слайд №4

4

Структурна схема програми

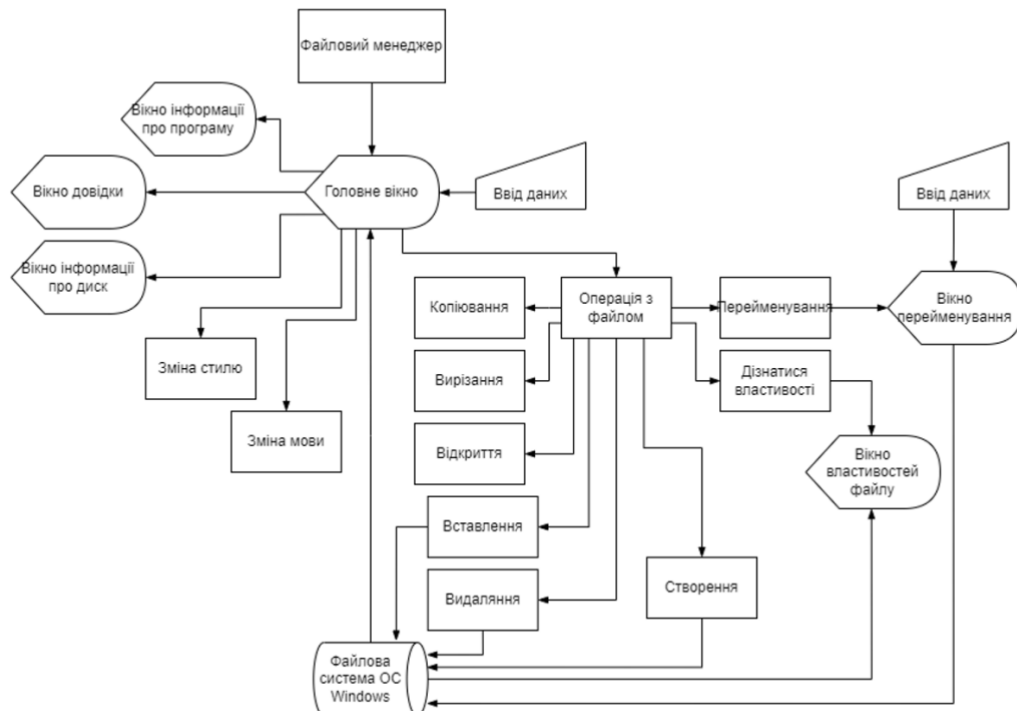


НУ "ЗП", Кафедра програмних засобів

5

Рисунок Б.5 – Слайд №5

Функціональна схема застосунку



НУ "ЗП", Кафедра програмних засобів

6

Рисунок Б.6 – Слайд №6

Рисунок Б.7 – Слайд №7

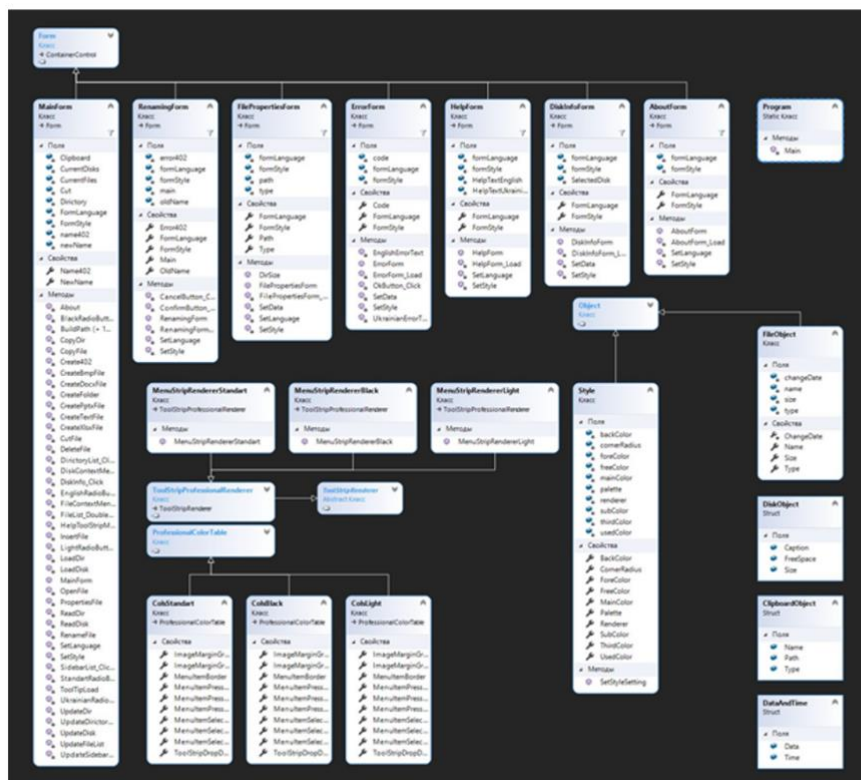
Назва	Опис	Використання
System	Містить фундаментальні та базові класи, що визначають типи значень і посилань, що часто використовуються, події та обробники подій, інтерфейси, атрибути та винятки обробки.	Дозволяє використовувати базові елементи.
ComponentModel	Надає класи, що використовуються для реалізації поведінки компонентів та елементів керування під час розробки та виконання. Цей простір імен містить базові класи та інтерфейси для реалізації атрибутів та перетворювачів типів, прив'язки до джерел даних та ліцензування компонентів.	Дозволяє взаємодіяти з інтерфейсом
Diagnostics	Надає класи, що дозволяють здійснювати взаємодію з системними процесами, журналами подій та лічильниками продуктивності.	Дозволяє викликати консоль
Drawing	Надає доступ до основних графічних функцій	Ручне малювання елементів форми
Guna.U12 WinForms	Пакет для створення інтерфейсу користувача	Додає більше можливостей для змінювання дизайну елементів форми

НУ "ЗП", Кафедра програмних засобів

7

Рисунок Б.7 – Слайд №7

Діаграма класів



НУ "ЗП", Кафедра програмних засобів

8

Рисунок Б.8 – Слайд №8

Створені зображення для файлів та папок

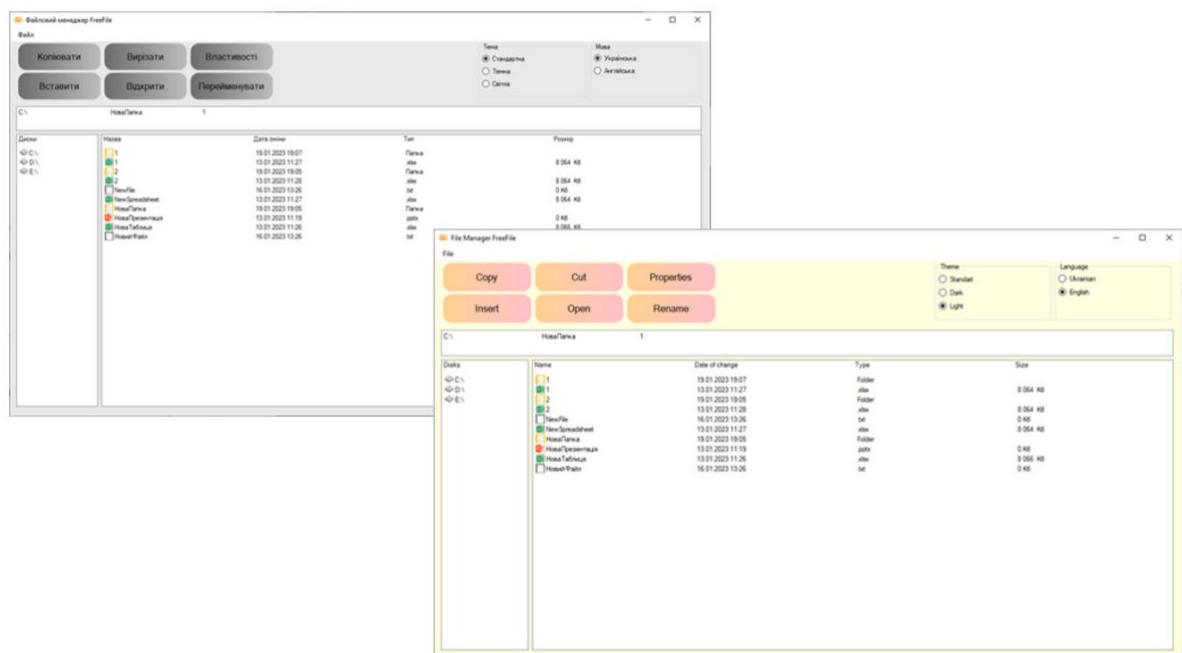


НУ "ЗП", Кафедра програмних засобів

11

Рисунок Б.11 – Слайд №11

Головне вікно файлового менеджера

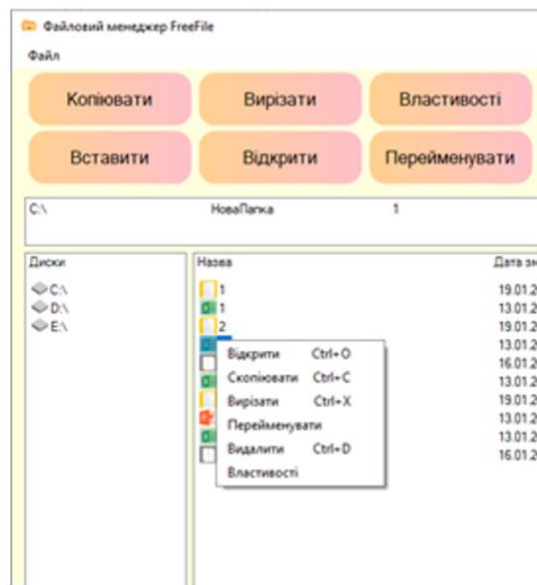


НУ "ЗП", Кафедра програмних засобів

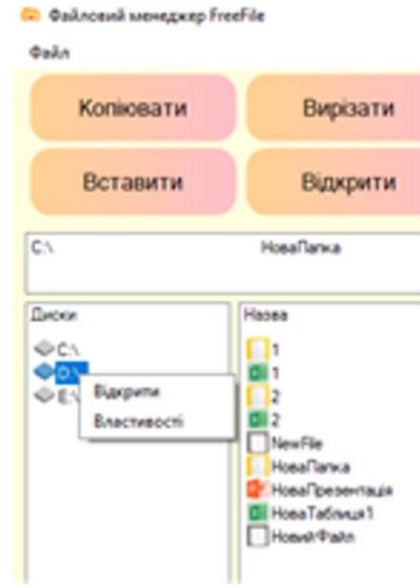
12

Рисунок Б.12 – Слайд №12

Меню для виділених об'єктів



Для файлів та тек



Для диску

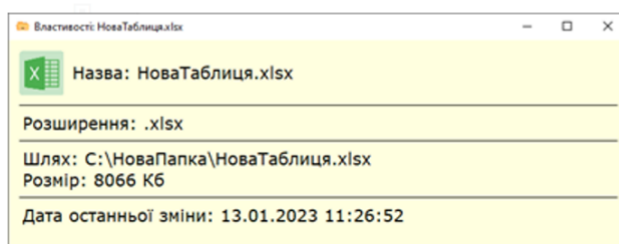


НУ "ЗП", Кафедра програмних засобів

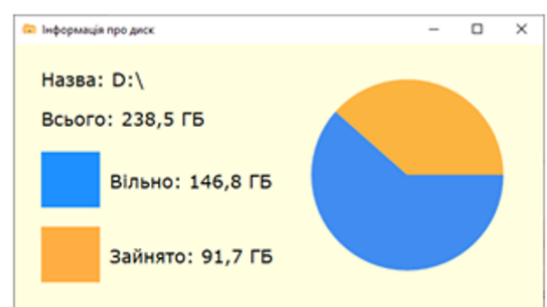
13

Рисунок Б.13 – Слайд №13

Приклади інформації, що надає файловий менеджер



Властивості обраного файлу
або теки



Інформація про диск



НУ "ЗП", Кафедра програмних засобів

14

Рисунок Б.14 – Слайд №14

Висновки

- проведено аналіз предметної області;
- проведено огляд програмних аналогів;
- розроблено технічне завдання;
- розроблено структуру програмного забезпечення;
- розроблено структуру класів програмного забезпечення;
- програмно реалізовано застосунок;
- розроблено необхідну документацію.



НУ "ЗП", Кафедра програмних засобів

Рисунок Б.15 – Слайд №15

15

**Доклад закінчено,
дякую за увагу!**



НУ "ЗП", Кафедра програмних засобів

Рисунок Б.16 – Слайд №16

16