

Міністерство освіти і науки України

Національний університет «Львівська політехніка»



## Курсовий проект

з дисципліни: «Системне програмне забезпечення»,

на тему: «Розробка програмного забезпечення для роботи з файловою системою»

Виконав:

ст. гр. КІ-306

Фодор А. Ю.

Перевірив:

Олексів М. В.

Львів – 2024

## **ЗАВДАННЯ**

Розробка програмного забезпечення для роботи з файловою системою. Розробка програми, що дозволяє керувати файловою системою на комп'ютері з операційною системою Windows.

## АНОТАЦІЯ

Результатом виконання цієї роботи є повноцінна функціональна системна утиліта - програма для керування файлами в ОС Windows. Основний функціонал цієї програми полягає в тому, щоб спростити роботу з файлами, зробити виконання різних операцій над ними зручними. Досягнуто бажаного результату за допомогою алгоритмів, в основі яких лежить використання системних викликів операційної системи Windows. У проекті розробив алгоритм відображення файлів та папок директорії та додаткової інформації про них, реалізував перехоплення системних виключень та їх обробку.

В даній записці до курсового проекту описана розробка утиліти - файловий менеджер, що включає в себе наступні можливості:

- Створення файлу/директорії
- Відкриття файлу/директорії
- Видалення файлу/директорії
- Перейменувати файл/директорію
- Копіювати файл/директорію
- Перегляд властивостей файлу
- Перемістити файл/директорію

Дана утиліта розроблена мовою програмування C++ з використанням вбудованих бібліотек.

## ЗМІСТ

ВСТУП .....	5
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД.....	6
1.1. Огляд файлових менеджерів та засобів їх проектування .....	6
1.2 Поняття і призначення файлових менеджерів .....	7
1.3 Огляд найпопулярніших файлових менеджерів.....	7
РОЗДІЛ 2. ПРОЄКТУВАННЯ.....	13
2.1 Аналіз задачі розробки файлового менеджера .....	13
2.2 Вибір технології програмування .....	15
2.3 Вибір засобів розробки.....	16
2.4 Організація роботи .....	17
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ .....	31
3.1. Основні функції програми .....	31
РОЗДІЛ 4. ТЕСТУВАННЯ.....	35
4.1. Інструкція для користувача .....	35
4.2. Тестування програми.....	36
4.3. Загальний висновок тестування програми .....	50
ВИСНОВКИ .....	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	52
ДОДАТОК .....	53

## ВСТУП

Файлова система є однією з ключових складових операційних систем, вона забезпечує організацію та управління файлами та каталогами. У сучасному цифровому світі, де обмін та збереження інформації має вирішальне значення, ефективна робота з файловою системою стає необхідністю для багатьох користувачів.

Цей курсовий проект присвячений розробці програмного забезпечення, спрямованого на поліпшення роботи з файловою системою операційної системи. Головною метою цього проекту є створення програми, яка надає користувачам зручний та інтерфейс для ефективної роботи з файлами та каталогами.

Задачі, поставлені перед цим проектом, включають розробку алгоритмів для створення, копіювання, переміщення та видалення файлів та папок, а також реалізацію функціоналу для перегляду інформації про файли. При розробці програми було використано мову програмування C++ та відповідні бібліотеки.

Очікується, що розроблена програма буде використовуватися користувачем для покращення роботи з файловою системою. Програма надасть можливість швидко та зручно виконувати операції з файлами, покращить організацію та навігацію по каталогах, а також сприятиме підвищенню продуктивності та ефективності роботи з інформацією.

У подальших розділах цієї курсової роботи будуть розглянуті деталі проекту та його реалізацію, включаючи архітектуру програми, методи і технології, використані при розробці, а також результати тестування та аналізу продуктивності. Все це дозволить отримати повну картину процесу розробки програмного забезпечення для роботи з файловою системою та його вплив на користувачів.

## **РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД**

У цьому розділі надається короткий опис проєкту та проводиться огляд методів існуючих розробок.

### **1.1. Огляд файлових менеджерів та засобів їх проектування**

Файловий менеджер є важливим інструментом у сфері обробки файлів і папок на операційній системі. Він надає користувачеві зручний спосіб взаємодії з файловою системою, дозволяючи створювати, копіювати, переміщати та видаляти файли та папки, а також здійснювати інші операції, пов'язані з управлінням файловою структурою.

При проектуванні файлових менеджерів ключовою метою є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу користувача. Вони повинні мати чітку ієрархічну структуру, де користувач може переходити по різних рівнях папок та файлів. Звичайно, основні функції, такі як копіювання, переміщення та видалення файлів, повинні бути легко доступними та інтуїтивно зрозумілими.

Одним з важливих аспектів проектування файлових менеджерів є розуміння роботи з файловою системою операційної системи. Це включає вивчення системних викликів, бібліотек та інших інструментів, які дозволяють взаємодіяти з файловою системою. Часто використовується API операційної системи, яке надає функції для виконання різних операцій з файлами та папками.

Огляд файлових менеджерів також включає аналіз існуючих рішень на ринку. Різні операційні системи мають власні вбудовані файлові менеджери, які можуть мати свої унікальні особливості. Крім того, існують сторонні файлові менеджери, які надають додаткову функціональність та переваги. Аналіз існуючих рішень допомагає зрозуміти потреби користувачів та визначити можливі шляхи для покращення інтерфейсу та функціональності.

## **1.2 Поняття і призначення файлових менеджерів**

Файловий менеджер - це програмний інструмент, який дозволяє користувачам працювати з файловою системою операційної системи. Він надає зручний інтерфейс для керування файлами і папками, виконання операцій з ними (створення, копіювання, переміщення, видалення) та перегляду вмісту директорій.

Файлові менеджери забезпечують користувачам можливість організації, навігації і керування файлами на комп'ютері. Вони дозволяють швидко знаходити, відкривати і редагувати файли, створювати нові папки та переміщати файли між різними директоріями. Деякі файлові менеджери також підтримують роботу з архівами, пошук файлів, попередній перегляд зображень і відтворення медіафайлів.

Файлові менеджери можуть мати різні функції і особливості, включаючи подвійну панель для одночасного відображення двох директорій, можливість роботи з мережевими ресурсами, підтримку різних операційних систем, розширені можливості налаштування та інші.

Основна функція файлових менеджерів - надати користувачам зручний спосіб керування файлами і папками на комп'ютері. Деякі з них також надають додаткові функції, що полегшують роботу з файлами і забезпечують більш продуктивний робочий процес.

## **1.3 Огляд найпопулярніших файлових менеджерів**

### **Norton Commander**

У Norton Commander виведення інформації про вміст каталогів здійснюється в так званих панелях. Панелі – це два прямокутні вікна, обмежені рамкою. Розмір цих вікон або фіксований і становить 40 \* 23 (повний розмір) і 40 \* 15 (половинний розмір) символів, або залежить від розмірів батьківського вікна (Windows Commander). Будь-яку панель можна вимкнути, а також можна прибрати всі панелі з екрану і залишити екран MS DOS. У Windows Commander такої можливості немає – в ньому кожен додаток запускається в окремому вікні.

Історично склалося, що панелі в Norton-подібних оболонках (виключаючи знову ж Windows Commander) виводяться у вигляді білих символів на синьому тлі. Саме це спочатку кидається в очі при використанні Norton-подібних оболонок. У Windows Commander ж в якості робочої палітри прийнята палітра Windows за замовчуванням: чорні символи на білому тлі. У більшості Norton - подібних оболонок існує можливість зміни палітри, використовуваної за замовчуванням. Для цього користуються утилітами конфігурування оболонок.

Панелі в Norton Commander мають, принаймні, 3 режими відображення інформації: коротка форма змісту, повна форма змісту і відображення дерева каталогів. У Volkov Commander до цих базових режимів відображення додається так звана інформаційна панель (з інформацією про пристрій, на якому розташований відображений на сусідній панелі каталог). У Norton Commander 3.0 крім цього є режим перегляду в сусідній панелі вмісту файлу, розташованого в поточній панелі. В інших версіях Norton Commander for DOS з'явилися панелі для перегляду вмісту архіву, паспорти каталогу й результату пошуку файлів. Проте найбільш "різноманітним" у використанні різних режимів відображення каталогів є оболонка Far.

Одна з панелей в Norton Commander завжди активна. В активній панелі відображається поточна директорія локального диску. Вона має виділений заголовок. Панель називається активною (або активізованою) тому, що вона має фокус вводу.

Управління режимами відображення панелей здійснюється окремо для правої і лівої панелі. Майже у всіх Norton - подібних оболонках (за винятком Dos Navigator і Windows Commander) існують окремі пункти системного меню для управління лівою і правою панелями. У DOS Navigator і Windows Commander для управління форматами панелей служить один пункт меню. Його вміст змінюється в залежності від того, яка панель активна.



Крім панелей, Norton Commander може містити так званий рядок міні-статусу. У цьому рядку містяться основні призначення функціональних клавіш-акселераторів клавіатури. На відміну від рядка стану вікон в WIMP - інтерфейсі цей рядок не є пасивним, з рядка міні-статусу можна мишкою викликати на виконання команди.

### **Total Commander**

Total Commander — один з найпотужніших двопанельних файлових менеджерів. На відміну від інших файлових менеджерів, у Total Commander не обмежуються лише функціями роботи з файлами і папками. TotalCommander представляє неабиякий асортимент численних додаткових опцій і необхідних інструментів, об'єднаних в єдину оболонку. Використовуючи стандартні можливості встановленої ОС задля досягнення тієї чи іншої мети (наприклад, установка з'єднання з FTP-сервером, копіювання інформації з мережі, групове і одночасне перейменування відразу кількох об'єктів, перегляд і редагування файлів та ін.) вам знадобиться звернутися до декількох додатків ОС. TotalCommander дозволить здійснити вищеописані операції просто обравши відповідну функцію у головному меню програми.

#### Основні можливості Total Commander:

- використання гарячих клавіш як в оригіналі DOS;
- підтримка перетягування об'єктів за допомогою функції Drag-and-Drop;
- розширене копіювання, переміщення, перейменування і видалення цілих директорій;
- обробка і перегляд архівів як субдиректорій папок;
- командний рядок для простого запуску програм з вказаними параметрами;

- розширена функція пошуку з повно-текстовим пошуком в будь-яких файлах між різними дисками;
- внутрішня функція розпакування ZIP-файлів без втручання зовнішніх утиліт;
- внутрішні розпаковщики архівів ZIP, ARJ, LZH, GZ, TAR, RAR і ACE форматів;
- внутрішній пакувальник ZIP-архівів;
- робота з мережевими дисками і FTP-серверами;
- повна підтримка Unicode в більшості функцій, де це можливо, включаючи FTP, вбудований ZIP-пакувальник і інтерфейс плагінів;
- файл довідки тепер за замовчуванням поставляється в HTML-форматі (CHM) ;
- захист збережених FTP-паролів за допомогою майстра-паролів, що використовує шифрування за алгоритмом AES256.

Отже, Total Commander - це швидкий і стабільний файловий менеджер, який має безліч можливостей для роботи з файлами та директоріями ОС.

### **Far Manager**

FAR Manager – це консольний файловий менеджер для операційних систем сімейства Windows. Ця програма виглядає як DOS - оболонка, але пристосована під Windows. Значна перевага цієї програми – це підтримка багатьох плагінів. Недолік програми полягає в її налаштуванні. На даний момент плагінів для FAR Manager існує дуже багато, що й зробило цю програму популярною. Наведемо деякі найважливіші складові програми:

- Архіватори в Far. Ви можете створювати, переглядати, змінювати архіви найбільш популярних форматів, таких як RAR, ZIP, ARJ, HA, CAB.

- Модулі Far:
- керування принтерами (локальними і мережевими);
- підсвічування синтаксису у вихідних текстах програм;
- робота з FTP-серверами (з підтримкою доступу через різні типи проксі, докачкою та ін.);
- пошук і заміна символів одночасно у групі файлів із застосуванням регулярних виразів;
- перейменування груп файлів з можливістю використання складних масок із символів підстановки і шаблонів;
- робота при нестандартних розмірах текстового екрану;
- перекодування текстів з урахуванням національних кодових таблиць;
- маніпуляції з вмістом кошика;
- автозавершення слів в редакторі і робота з шаблонами;
- редагування системного реєстру Windows;
- робота з різними серверами через ODBC;
- відображення вмісту файлів довідки Windows;
- калькулятори з різними можливостями функції перевірки орфографії при обробці тексту в редакторі FAR;
- підготовка каталогу змінних накопичувачів.

## **XYplorer**

XYplorer володіє багатьма розширеними функціями, що полегшують роботу з файлами. Він надає можливість швидкого пошуку файлів, керування закладками, перегляду дерева папок і редагування файлових атрибутів. XYplorer також підтримує шифрування файлів і має інструменти для продуктивної роботи з файловою системою.

## **Windows Explorer**

Windows Explorer є вбудованим файловим менеджером в операційну систему Windows. Він надає базові функції керування файлами, такі як копіювання, переміщення, видалення та перегляд вмісту папок. Windows Explorer є стандартним інструментом для багатьох користувачів Windows і використовується для повсякденних завдань з роботою з файлами.

Кожен файловий менеджер має свої переваги та особливості, і вибір залежить від конкретних потреб користувача. Важливо обрати файловий менеджер, який найкраще задовольнить ваші вимоги щодо функціональності та зручності використання.

Отже, після проведеного порівняльного аналізу можна зробити висновок, що найпопулярнішою та найбільш функціональною утилітою для роботи з файловою системою є Total Commander.

## РОЗДІЛ 2. ПРОЄКТУВАННЯ

### 2.1. Аналіз задачі розробки файлового менеджера

У розділі "Проектування" курсової роботи з фокусом на розробку програмного забезпечення для роботи з файловою системою, розглядається процес планування, архітектури та розробки файлового менеджера. Цей розділ детально описує етапи проектування, включаючи визначення вимог, проектування інтерфейсу користувача та розробку функціональності.

**1. Визначення вимог:** Функціональні вимоги включають створення файлів та директорій, відкриття, видалення, перейменування та переміщення файлів та директорій, а також копіювання файлів та перегляд їх властивостей.

Визначаються нефункціональні вимоги, такі як продуктивність, надійність, зручність використання та підтримка операційної системи Windows.

**2. Архітектура програми:** Для розробки даного файлового менеджера використовується мова програмування C++ та системна бібліотека операційної системи Windows - WinApi. Ці технології дозволяють ефективно взаємодіяти з файловою системою через консольний інтерфейс.

Файловий менеджер буде розроблено за архітектурою клієнт-сервер. Взаємодія з файловою системою відбуватиметься через системні виклики WinApi, які виконуватимуться на рівні серверної частини. Клієнтська частина буде відповідальна за взаємодію з користувачем через консольний інтерфейс.

### **3. Розробка функціональності:**

**Створення файлів та директорій:** Розробляються алгоритми для створення нових файлів та директорій в заданому розташуванні.

**Відкриття файлів та директорій:** Розробляються методи для відкриття файлів та директорій з подальшою можливістю редагування чи перегляду вмісту.

**Видалення файлів та директорій:** Розробляються алгоритми для безпечного

видалення файлів та директорій з можливістю підтвердження операції.

Перейменування файлів та директорій: Розробляються методи для перейменування файлів та директорій з урахуванням валідації нових назв.

Копіювання файлів та директорій: Розробляються алгоритми для копіювання файлів та директорій з можливістю обробки конфліктів.

Перегляд властивостей файлів: Розробляються методи для отримання та відображення властивостей файлів, таких як розмір, тип та дата останньої зміни.

Переміщення файлів та директорій: Розробляються алгоритми для переміщення файлів та директорій між різними розташуваннями.

**4. Тестування та налагодження:** Розробляється план тестування, який включає модульні тести для окремих функцій, інтеграційні тести для перевірки взаємодії компонентів та прийняття функціональних тестів для перевірки відповідності вимогам.

Також розробляється процес виявлення та виправлення помилок під час розробки та тестування файлового менеджера.

**5. Документування:** Розробляється документація, яка включає опис функціональності, архітектури, вимог, процесу розробки, керівництва користувача та інші необхідні матеріали.

Забезпечується належне коментування коду, що допоможе зрозуміти логіку програми та підтримати її в майбутньому.

**6. Розгортання та експлуатація:** Розробляється процес підготовки утиліти до розгортання, включаючи створення необхідних інсталяційних файлів та документації. Також розробляється керівництво користувача, яке надає інструкції щодо встановлення та використання файлового менеджера.

## 2.2. Вибір технології програмування

Використання WinApi дозволяє прямо взаємодіяти з операційною системою Windows, надаючи доступ до різних системних функцій та можливостей файлової системи. Це дозволить ефективно працювати з файлами та директоріями, виконувати операції створення, копіювання, переміщення та видалення.

Мова C++ відома своєю швидкістю та ефективністю, що є важливими факторами для файлових менеджерів, особливо при роботі з великими обсягами даних або операціями з файлами.

Використання мови C++ дозволяє використовувати різні бібліотеки, які спрощують розробку файлових менеджерів. Наприклад, можна використовувати бібліотеки для роботи з файловою системою, роботи з інтерфейсом користувача або для додаткових функцій, які поліпшують досвід користувача.

Використання WinApi забезпечує переносимість між різними версіями операційної системи Windows, що дозволяє вашому файловому менеджеру працювати на різних комп'ютерах з різними операційними системами.

Використання мови C++ та WinApi надає доступ до розширених можливостей операційної системи, таких як обробка системних виключень, робота з потоками та багатопоточність, безпека файлів та багато іншого.

Загалом, використання мови програмування C++ з WinApi є відмінним вибором для розробки файлового менеджера, оскільки ця технологія надає прямий доступ до системних функцій та забезпечує швидкість, ефективність та розширюваність вашого проекту.

### 2.3. Вибір засобів розробки

Для виконання поставленого завдання використано мову програмування C++ та середовище програмування Microsoft Visual Studio 2022.

Вибір мови програмування можна обґрунтувати тим, що мова C++ має у своєму складі безліч класів та методів для роботи з файловою системою, процесами та потоками ОС Windows. У цієї мови є багато можливостей для реалізації найскладніших алгоритмів і роботи з операційною системою та апаратним середовищем.

Середовище програмування Microsoft Visual Studio 2022 має ряд переваг, таких як: Підтримка мов програмування: Visual Studio підтримує широкий спектр мов програмування, включаючи C++ та інші. Це дає можливість вибрати мову, яка найкраще відповідає потребам проекту з розробки файлового менеджера.

Інтегроване середовище розробки: Visual Studio надає зручне та ефективне робоче середовище, яке дозволяє розробникам зосередитись на кодуванні. Вона має потужний редактор коду з підсвічуванням синтаксису, автодоповненням, вбудованими інструментами для налагодження та багато іншого.

Розширюваність: Visual Studio має велику кількість розширень та плагінів, які дозволяють розширити функціональність та забезпечити підтримку специфічних потреб розробки файлового менеджера. Наприклад, можна використовувати розширення для роботи з файловою системою, додаткових інструментів для взаємодії з операційною системою та інших корисних функцій.

Налагодження та профілювання: Visual Studio надає потужні інструменти для налагодження та профілювання програм. Це дозволяє розробникам виявляти та усувати помилки, а також оптимізувати продуктивність файлового менеджера.

Підтримка версійного контролю: Visual Studio має вбудовану підтримку різних систем контролю версій, таких як Git, SVN, TFS та інші. Це дозволяє зручно керувати версіями коду та співпрацювати з іншими розробниками.



Документація та спільнота: Visual Studio має широку документацію, навчальні матеріали та активну спільноту розробників, які можуть надати підтримку та відповісти на запитання під час розробки файлового менеджера.

Загалом, використання Visual Studio для розробки файлового менеджера надає розробникам зручність, продуктивність та доступ до різноманітних інструментів і ресурсів, що сприяють успішній реалізації проекту.

## **2.4. Організація роботи**

Тут описано підхід до створення великих проектів та обґрунтовано обраний підхід на фоні з іншими існуючими

### **1. Створення проекту**

Створення будь якого програмного продукту, в моєму випадку системна утиліта "файловий менеджер", часто вимагає спільної роботи команди фахівців. Це включає програмістів, тестувальників, дизайнерів, аналітиків і менеджерів проектів. Спільна робота в команді дозволяє ефективно розподілити завдання, використовувати різні навички та досвід кожного учасника, а також забезпечує контроль якості на кожному етапі розробки.

### **2. Організація роботи команди**

Правильна організація роботи команди є критично важливою для успіху будь-якого проекту. Це включає планування, визначення ролей і відповідальностей, встановлення чітких цілей і термінів, а також комунікацію між учасниками команди. Добре організована команда здатна працювати більш продуктивно, ефективно вирішувати проблеми і своєчасно завершувати проект.

### **3. Підходи та методології**

Для організації роботи команди існує багато різних методологій та підходів. Основні:

## **Waterfall**

Вотерфолл — це традиційний підхід до розробки програмного забезпечення, який передбачає послідовне виконання етапів розробки. Проект розділяється на чітко визначені фази: аналіз вимог, дизайн, розробка, тестування, впровадження та підтримка. Кожна фаза повинна бути завершена перед тим, як розпочнеться наступна. Цей підхід підходить для проектів з чітко визначеними вимогами та стабільним середовищем.

## **Agile**

Аджайл — це гнучкий підхід до розробки програмного забезпечення, який дозволяє адаптуватися до змін вимог та умов. Команда працює в коротких ітераціях (спринтах), кожна з яких завершується випуском працюючого продукту або його частини. Аджайл передбачає тісну співпрацю з замовником, регулярні зустрічі для обговорення прогресу та швидке реагування на зміни.

## **Kanban**

Канбан — це метод управління проектами, який фокусується на візуалізації процесу роботи та оптимізації потоку завдань. Він використовує канбан-дошку з колонками, що представляють різні етапи роботи (наприклад, "Виконати", "В процесі", "Готово"). Завдання переміщуються по дошці відповідно до свого статусу. Канбан допомагає командам контролювати обсяг роботи, уникати перевантажень і забезпечувати безперервний потік виконання завдань.

## **4. Можливості GitHub**

У моєму проекті для організації роботи використовується GitHub Issues та GitHub Projects. Ці інструменти дозволяють створювати, відстежувати та керувати завданнями, що забезпечує прозорість та ефективність у командній роботі.

### **GitHub Issues**

GitHub Issues дозволяє створювати завдання, баги та запити на покращення. Кожне Issue може містити детальний опис, коментарі, мітки (labels), призначення

виконавців (assignees) та терміни виконання (milestones). Це допомагає тримати всі завдання організованими та пріоритезованими.

### **GitHub Projects**

GitHub Projects — це інструмент для управління проектами, який дозволяє створювати канбан-дошки для візуалізації та управління завданнями. Завдання з GitHub Issues додані до дошки, що допомагає відстежувати прогрес і забезпечує видимість статусу кожного завдання.

### **User Stories**

User Stories (користувацькі історії) — це інструмент, що використовується для опису функціональних вимог з точки зору кінцевого користувача. Кожна User Story зазвичай має формат "Як [користувач], я хочу [функціональність], щоб [користь]". Це допомагає команді зрозуміти потреби користувачів і фокусуватися на створенні цінності для них.

## **5 .Висновок**

Використання правильних методологій та інструментів для організації роботи команди є ключем до успішної розробки програмного продукту. Вибір підходящого методу, такого як Waterfall, Agile або Kanban, а також використання інструментів, таких як GitHub Issues та Projects, допомагає забезпечити ефективність, продуктивність і якість роботи команди.

## **6. Реалізація GitHub Projects на моєму проєкті**

У своєму проєкті я створив необхідні user stories як картки issues та додав їх до двох проєктів GitHub Projects.

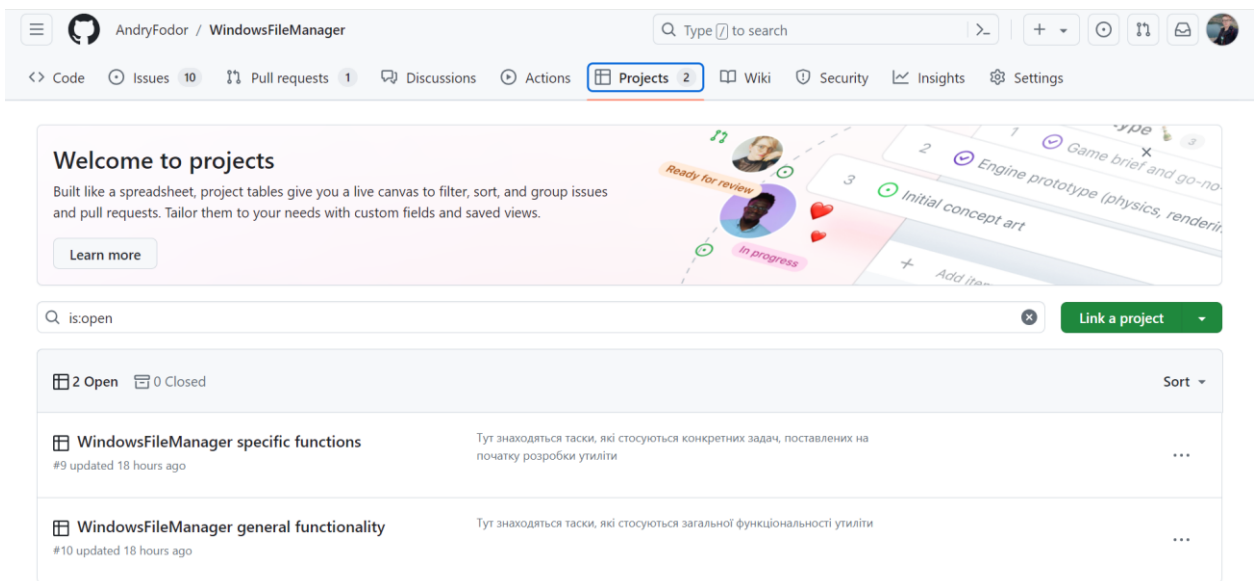


Рис. 1. Два GitHub projects на моєму проекті

Перший проект має назву [WindowsFileManager specific functions](#), другий - [WindowsFileManager general functionality](#)

В кожному проекті зберігаються issues з конкретними user stories. Це можна побачити на рисунку 2

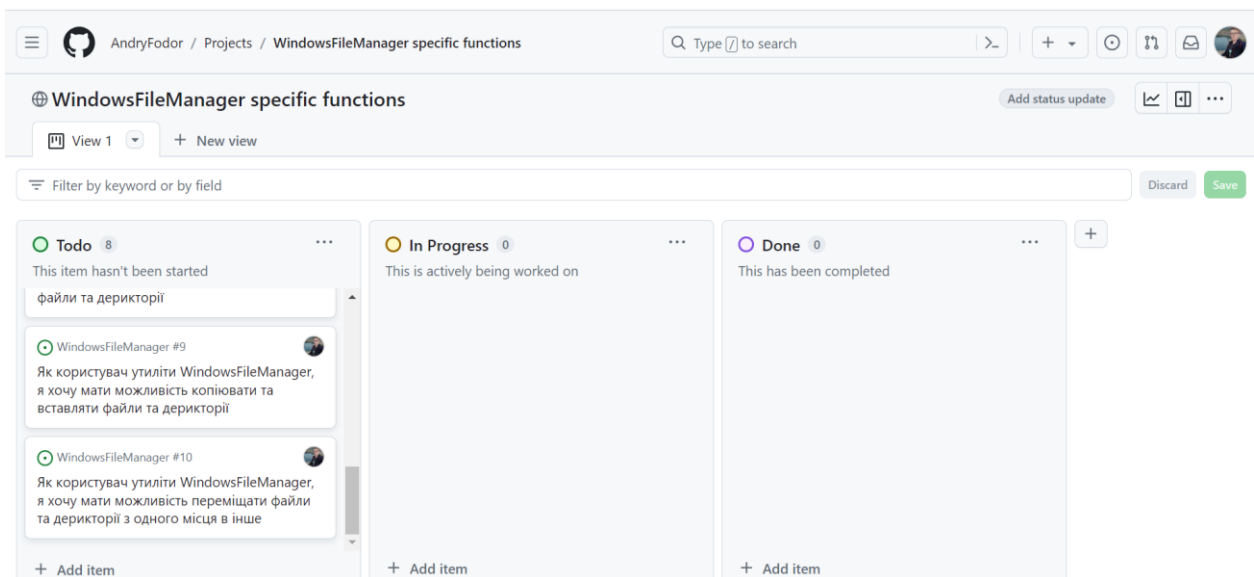


Рис. 2. Проект WindowsFileManager specific functions

Як можна побачити, цей інструмент дозволяє організовувати роботу в стилі Kanban таблиць. Коли розробник, який має виконати відповідне завдання (воно позначене на ньому), починає над ним працювати, він перетягує відповідну картку в таблицю In Progress. Завдання знаходиться тут до того часу, поки розробник не закінчить працювати над відповідною задачею. Коли завдання виконане та відтестоване, розробник пересуває картку в Done. Таким чином проєкт менеджери можуть легко відслідковувати роботу команди та керувати нею.

## 7. Реалізація user stories як issues на моєму проєкті

Для того, щоб переглянути всі issues, можна перейти у однойменну вкладку в репозиторії і переглянути їх. Також, якщо вони прикріплені до якихось конкретних проєктів, їх можна знайти там. Виглядає вкладка Issues так, як

зображено на рисунку 3. Якщо переглянути деталі конкретної задачі, можна побачити заголовок, опис, лейбли, мейлстоуни, кому призначена відповідні задача, до якого проєкту відпоситься, де розробляється (в якій гілці) та в якому стані виконання знаходиться, коментарі та історію її зміни.

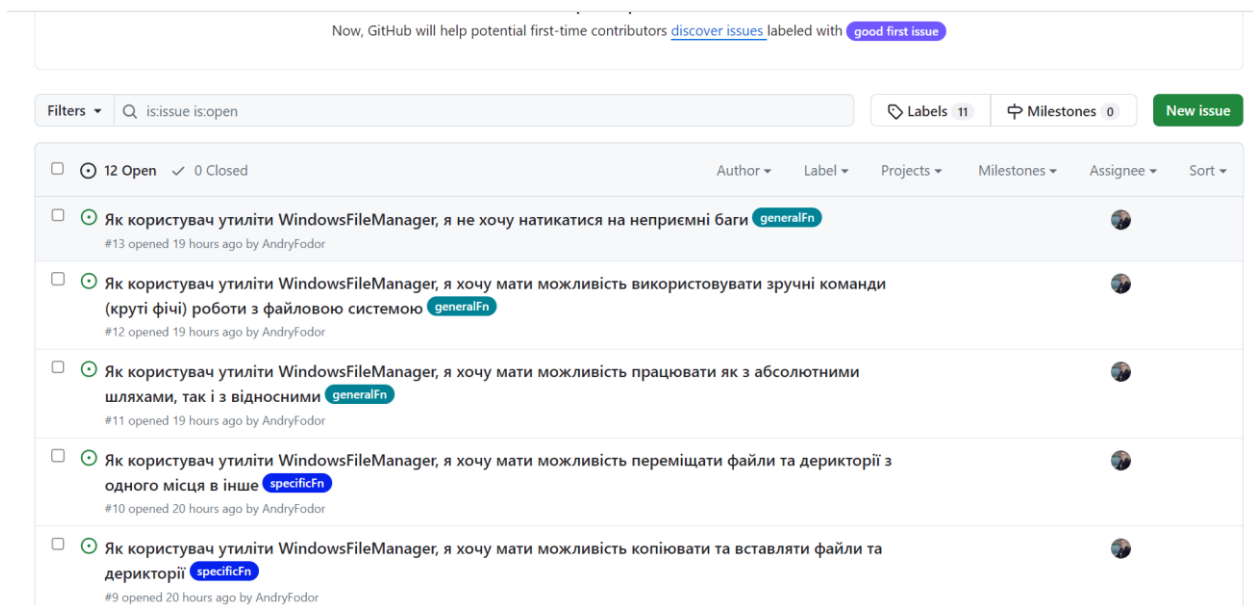



Рис. 3. Вкладка Issues

## Issues 1

### Меню

Як користувач утиліти WindowsFileManager, я хочу бачити можливості даного продукту у вигляді меню #2

[Open](#) AndryFodor opened this issue yesterday · 0 comments



AndryFodor commented yesterday

Owner

Обсяг робіт

- створити набір можливостей утиліти як структура даних типу array, vector etc.
- детально продумати вигляд меню (зручний вивід поточної локації користувача та список можливостей)
- створити функцію, яка прийматиме необхідні параметри та виводитиме меню користувачу

Естімейт

7 годин

AndryFodor self-assigned this yesterday

Assignees

AndryFodor

Labels

generalFn

Projects

WindowsFileManager general functional...

Status: Todo

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.


Рис. 4. Issue 1

## Issues 2

### Вміст поточної дерикторії

Як користувач WindowsFileManager, я хочу, щоб в мене була можливість бачити вміст поточної дерикторії #3

[Open](#) AndryFodor opened this issue yesterday · 0 comments



AndryFodor commented yesterday

Owner

обсяг робіт

- перевірити, чи пуста поточна дерикторія чи ні
  - пуста? вивести про це один вид повідомлення
  - інакше: сформулювати правильний вигляд зручного перегляду контенту
- перебрати весь вміст дерикторії
- визначити тип поточного елемента дерикторії (папка чи файл) та вивести його
- визначити ім'я поточного елемента дерикторії та вивести його
- визначити час останньої зміни відповідного елемента дерикторії та вивести це значення

Естімейт

10 год

1

Assignees

AndryFodor

Labels

specificFn

Projects

WindowsFileManager specific functions

Status: Todo

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

Рис. 4. Issue 2

## Issues 3

### Видаляти папки та файли

Як користувач утиліти WindowsFileManager, я хочу мати можливість ефективно видаляти папки та файли #4


 Open AndryFodor opened this issue yesterday · 1 comment

Рис. 5. Issue 3

## Issues 4

### Створювати файли та папки

Як користувач утиліти WindowsFileManager, я хочу мати можливість зручно створювати файли та папки #5

 Open AndryFodor opened this issue yesterday · 0 comments


Рис. 6. Issue 4

## Issues 5

### Отримання властивостей файлу

Як користувач утиліти WindowsFileManager, я хочу мати можливість отримувати властивості файла #6

 Open AndryFodor opened this issue yesterday · 0 comments

**AndryFodor** commented yesterday

Owner ...

**Обсяг робіт**


Створити функцію, яка буде приймати шлях до відповідного файлу та виводити його властивості в зручному для користувача вигляді. Для цього слід:

- перевірити, чи існує файл за отриманим шляхом
  - якщо не існує, не виконувати жодних операцій
  - вивести повідомлення про некоректний шлях до файлу
- якщо файл існує, сформулювати зручний формат виводу інформації
- отримати повний шлях до файлу і вивести його
- отримати розмір файлу і вивести його
- отримати дозволи файлу та вивести їх

**Естимейт**

3 години


Assignees

 AndryFodor

Labels

**specificFn**

Projects

 WindowsFileManager specific functions

Status: Todo ▾

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.


Рис. 7. Issue 5

## Issues 6

### Відкривати файли і бачити їхній вміст

Як користувач утиліти WindowsFileManager, я хочу мати можливість відкривати файли і бачити їхній вміст #7

 Open AndryFodor opened this issue 20 hours ago · 0 comments

**AndryFodor** commented 20 hours ago

Owner ...

**Обсяг робіт**


Створити функцію, яка буде отримувати шлях до файлу, читати його вміст та виводити. Для цього:

- виконати необхідні перевірки (коректність шляху, можливість відкриття файлу)
- у випадку помилок, опрацювати їх належним чином, вивівши про це повідомлення
- визначити зручний для користувача вивід контенту файлу
- відкрити файл
- зчитати весь його контент та вивести користувачу
- закрити файл

**Естимейт**

5 годин


Assignees

 AndryFodor

Labels

**specificFn**

Projects

 WindowsFileManager specific functions

Status: Todo ▾

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

Рис. 8. Issue 6




## Issues 7

### Перейменовувати файли та дерикторії

Як користувач утиліти WindowsFileManager, я хочу мати можливість перейменовувати файли та дерикторії #8

Edit New issue

Open AndryFodor opened this issue 20 hours ago · 0 comments



AndryFodor commented 20 hours ago

Owner

Обсяг робіт

Для реалізації поставленої задачі слід створити функцію, яка буде приймати параметрами шлях до файлу чи дерикторії, які треба перейменувати, та нове ім'я. Для реалізації логіки слід дотримуватися таких пунктів:

- виконати перевірку коректності отриманого шляху. Опрацювати належним чином ситуацію, коли передано неправильний шлях:
  - не змінювати нічого в структурі
  - вивести повідомлення про некоректність шляху
- на основі отриманого шляху та імені перевірити, чи не існує вже файла чи дерикторії з таким іменем для уникнення конфліктів в системі. Для цього:
  - виконати необхідну перевірку
  - якщо файл чи дерикторія вже існує, не перейменовувати вказаний файл чи дерикторію
  - вивести повідомлення про неможливість виконання операції та пояснити причину
  - закінчити функцію
- в протилежному випадку виконати операцію перейменування

Естимейт

7 годин

Assignees

AndryFodor

Labels

specificFn

Projects

WindowsFileManager specific functions

Status: Todo

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

Notifications

Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

Рис. 9. Issue 7


## Issues 8

### Копіювати та вставляти файли та дерикторії

Як користувач утиліти WindowsFileManager, я хочу мати можливість копіювати та вставляти файли та дерикторії #9

Edit New issue

Open AndryFodor opened this issue 20 hours ago · 0 comments



AndryFodor commented 20 hours ago

Owner

Обсяг робіт

Для досягнення очікуваного результату створити функцію, яка буде приймати шлях до файлу чи дерикторії, яку слід скопіювати, та шлях, де користувач хоче бачити копію відповідного елемента. Для цього слід:

- підготувати зручний для користувача вивід
- перевірити коректність отриманих шляхів
- у випадку некоректності вивести про це повідомлення і закінчити виконання функції
- у випадку коректності виконати операцію копіювання файлу чи дерикторії
- відловити можливі помилки при виконанні операції та вивести їх користувачу

Естимейт

4 години

Assignees

AndryFodor

Labels

specificFn

Projects

WindowsFileManager specific functions

Status: Todo

Milestone

No milestone

Development

Рис. 10. Issue 8


## Issues 9

### Переміщати файли та дерикторії з одного місця в інше

Як користувач утиліти WindowsFileManager, я хочу мати можливість переміщати файли та дерикторії з одного місця в інше #10

Edit New issue

Open AndryFodor opened this issue 20 hours ago · 0 comments



AndryFodor commented 20 hours ago

Owner

### Обсяг робіт

Створити функцію, яка буде приймати шлях до файлу чи дерикторії, які слід перемістити, та шлях, куди їх треба перемістити та виконати необхідну логіку. Для цього:

- підготувати зручний для користувача вивід інформації
- перевірити коректність отриманих параметрами шляхів
- у разі некоректності вивести про це повідомлення та завершити виконання функції
- в протилежному випадку спробувати виконати необхідну операцію
- відловити можливі помилки під час виконання операції та вивести їх користувачу

### Естимейт

4 години

Assignees

AndryFodor

Labels

specificFn

Projects

WindowsFileManager specific functions

Status: Todo

Milestone

No milestone

Development

Рис. 11. Issue 9

## Issues 10

### Працювати як з абсолютними шляхами, так і з відносними

Як користувач утиліти WindowsFileManager, я хочу мати можливість працювати як з абсолютними шляхами, так і з відносними #11

Edit New issue

Open AndryFodor opened this issue 20 hours ago · 0 comments



AndryFodor commented 20 hours ago

Owner ...

#### Обсяг робіт

Як говорить user experience, користувачу зручніше працювати з відносними шляхами в системі. Хоча всі функції працюють з абсолютними шляхами. В такому випадку, слід додати підтримку відносних шляхів в утиліті. Для цього можна дотримуватися таких пунктів:

- зберігати поточний шлях користувача в певній змінній. Ініціалізаційне значення має бути "D:\"
- виконати перевірку, чи існує цей диск
  - якщо існує, продовжити роботу
  - якщо не існує, присвоїти значення "C:\"
  - оскільки такий шлях існує завжди, додаткових перевірок не вимагається
- після кожного нового введення шляху користувачем, виконувати перевірку, чи абсолютний шлях, чи відносний
- у випадку, якщо він абсолютний, передати його у відповідну функцію без змін
- у випадку, якщо він відносний, необхідно перетворити його на абсолютний перед передачею у відповідну функцію. Для цього:
  - створити допоміжну функцію, яка буде приймати параметрами поточний абсолютний шлях та введений користувачем відносний
  - виконати перевірку коректності шляхів. Якщо шлях не існує, вивести про це повідомлення
  - якщо все добре, виконати конкатенацію двох шляхів та повернути результат
- передати в необхідну функцію відформатований абсолютний шлях.
- У випадку зміни поточної дерикторії, аналогічним способом опрацювати шлях. Далі:
- перевірити, чи можна виконати перехід до відповідної дерикторії
- у випадку некоректності шляху до необхідної дерикторії вивести про це повідомлення користувачу
- у випадку введення коректного шляху, змінити дерикторію, тобто:
  - вивести повідомлення про успішну зміну поточного місця розташування та нове місце розташування
  - перезаписати значення змінної, в якій зберігається поточне місце знаходження користувача
- далі використовувати цю змінну ідентично

#### Естимейт

14 годин



Assignees

AndryFodor

Labels

generalFn

Projects

WindowsFileManager general functional...

Status: Todo

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

Notifications

Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

1 participant



Lock conversation

Pin issue

Transfer issue

Delete issue

Рис. 12. Issue 10

## Issues 11

### Зручні команди (круті фічі) роботи з файловою системою

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links to Code, Issues (12), Pull requests (1), Discussions, Actions, Projects (2), Wiki, Security, Insights, and Settings. Below this, the issue title is 'Як користувач утиліти WindowsFileManager, я хочу мати можливість використовувати зручні команди (круті фічі) роботи з файловою системою #12'. It's marked as 'Open' and was opened by AndryFodor 20 hours ago with 0 comments. The issue body contains a section 'Обсяг робіт' (Scope of work) with a detailed description of the goal: to add support for convenient commands like 'cd' and 'ls' to the WindowsFileManager utility. A bulleted list outlines the tasks: ensuring command recognition, enabling path selection, using a modular support system, updating documentation, and testing. The 'Estimate' (Естимейт) is 10 hours. On the right, the 'Assignees' section lists AndryFodor. The 'Labels' section has a 'generalFn' label. The 'Projects' section shows 'WindowsFileManager general functional...' with a status of 'Todo'. The 'Milestone' section is empty. The 'Development' section has a link to 'Create a branch for this issue or link a pull request.' The 'Notifications' section has a 'Customize' link.

Рис. 13. Issue 11

## Issues 12

### Баги

### Як користувач утиліти WindowsFileManager, я не хочу натикатися на неприємні баги #13

Open AndryFodor opened this issue 20 hours ago · 0 comments

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links to Code, Issues (12), Pull requests (1), Discussions, Actions, Projects (2), Wiki, Security, Insights, and Settings. Below this, the issue title is 'Як користувач утиліти WindowsFileManager, я не хочу натикатися на неприємні баги #13'. It's marked as 'Open' and was opened by AndryFodor 20 hours ago with 0 comments. The issue body contains a section 'Обсяг робіт' (Scope of work) with a detailed description of the goal: to ensure the WindowsFileManager utility is free of bugs. A bulleted list outlines the tasks: testing each function, documenting bugs, updating documentation, testing the fix, and testing the final version. The 'Estimate' (Естимейт) is 1 hour for the function and 3 hours for the final testing. On the right, the 'Assignees' section lists AndryFodor. The 'Labels' section has a 'generalFn' label. The 'Projects' section shows 'WindowsFileManager general functional...' with a status of 'Todo'. The 'Milestone' section is empty. The 'Development' section has a link to 'Create a branch for this issue or link a pull request.' The 'Notifications' section has a 'Customize' link and an 'Unsubscribe' button.

Рис. 14. Issue 12

## **8. Реалізація окремих гілок розробки, прив'язаних до відповідних issues, на моєму проекті**

Для зручної роботи в команді створюються окремі гілки розробки. Зазвичай вони створюються не на основній гілці, а на деякій окремій гілці, яку можна буде протестувати і подивитися загальний результат роботи. В моєму випадку ця гілка називається `develop`. Ця гілка відходить з `main` та має на своїй основі всі решта гілок розробки мого проекту. Таким чином, після закінчення проекту на цій гілці можна буде побачити прототип готового продукту. Але можна буде легко протестувати його, знайти баги і виправити їх у випадку їх приступності ще в режимі розробки, а не продакшна. Відтестований продукт, який пройшов всі рев'ю, можна зливати з основною гілкою розробки та закінчувати проект. Приблизна схема мого дерева розробки має наступний вигляд (рис. 15). Як можна побачити, кожна гілка розробки йде паралельно одна одній, що дозволяє програмістам ефективно працювати в команді та швидко створювати майбутній продукт. Керування цим процесом належить проджект менеджерам, вони в свою чергу використовують різноманітні інструменти, такі як Trello, Jira, GitHub Pages & Issues (як в моєму випадку). Після закінчення розробки (а саме після закінчення створення цього звіту), буде перевірена та злита ця гілка до `develop`, виконається відповідне тестування, якщо воно потрібне, рев'ю готового продукту, та у разі, що все працює чудово, виконається пулл реквест в `main` гілку.

WindowsFileManager: --all - gitk

File Edit View Help

Commit Message	Author	Date
Start to prepare a general r	Andry <andrijfodor@gmail.c	2024-06-02 22:00:31
Merge pull request #24 from AndryFodor/r	Andry <122674916+AndryF	2024-06-02 21:57:05
fix dir creation. Add testing report	Andry <andrijfodor@gmail.c	2024-06-02 21:54:29
feat: +relative pathes, +menu, +cd&ls	Andry <andrijfodor@gmail.c	2024-06-02 19:39:25
Merge pull request #23 from AndryFodor/creatingOption	Andry <122674916+AndryF	2024-06-02 19:23:08
Merge branch 'develop' into creatingOption	Andry <122674916+AndryF	2024-06-02 19:22:57
Merge pull request #22 from AndryFodor/deletingOption	Andry <122674916+AndryF	2024-06-02 19:21:01
Merge branch 'develop' into deletingOptio	Andry <122674916+AndryF	2024-06-02 19:20:51
Merge pull request #21 from AndryFodor/fileProp	Andry <122674916+AndryF	2024-06-02 19:15:02
Merge branch 'develop' into fileProp	Andry <122674916+AndryF	2024-06-02 19:14:49
Merge pull request #20 from AndryFodor/dirContent	Andry <122674916+AndryF	2024-06-02 19:12:47
Merge branch 'develop' into dirContent	Andry <122674916+AndryF	2024-06-02 19:11:37
Merge pull request #19 from AndryFodor/moveOption	Andry <122674916+AndryF	2024-06-02 18:57:58
Merge branch 'develop' into moveOp	Andry <122674916+AndryF	2024-06-02 18:57:46
Merge pull request #18 from AndryFodor/fileOpening	Andry <122674916+AndryF	2024-06-02 18:52:41
Merge branch 'develop' into fileOp	Andry <122674916+AndryF	2024-06-02 18:52:31
Merge pull request #17 from AndryFodor/renameOpt	Andry <122674916+AndryF	2024-06-02 18:48:25
Merge branch 'develop' into ren	Andry <122674916+AndryF	2024-06-02 18:47:52
Merge pull request #16 from AndryFodor/copyDir	Andry <122674916+AndryF	2024-06-02 18:39:20
feat: + copyDirectory	Andry <andrijfodor@gmail.c	2024-06-02 18:24:38
feat: + copyFile	Andry <andrijfodor@gmail.c	2024-06-02 18:23:36
Merge pull request #14 from AndryFodor/requirements	Andry <122674916+AndryF	2024-06-02 18:38:14
Added	Andry <andrijfodor@gmail.c	2024-06-02 14:40:39
Merge pull request #15 from AndryFodor/userStories	Andry <122674916+AndryF	2024-06-02 18:37:51
Added use	Andry <andrijfodor@gmail.c	2024-06-02 16:55:22
feat: + renameDirectory fn	Andry <andrijfodor@gmail.c	2024-06-02 18:17:16
feat: + renameFile fn	Andry <andrijfodor@gmail.c	2024-06-02 18:14:36
feat: + openFile	Andry <andrijfodor@gmail.c	2024-06-02 18:01:55
feat: + moveDirectory fn	Andry <andrijfodor@gmail.c	2024-06-02 18:27:33
feat: + moveFile fn	Andry <andrijfodor@gmail.c	2024-06-02 18:26:58
Implement displayDirectoryContents fn	Andry <andrijfodor@gmail.c	2024-06-02 17:24:55
Implement displayFileProperties fn	Andry <andrijfodor@gmail.c	2024-06-02 17:15:19
Add directories deleting	Andry <andrijfodor@gmail.c	2024-06-02 17:38:46
add file deliting	Andry <andrijfodor@gmail.c	2024-06-02 17:36:42
Add file creating fn	Andry <andrijfodor@gmail.c	2024-06-02 17:51:23
Implement posibility to create directories	Andry <andrijfodor@gmail.c	2024-06-02 17:43:39
Project initialization	Andry <andrijfodor@gmail.c	2024-06-02 14:01:20
Merge pull request #1 from AndryFodor/devel	Andry <122674916+AndryF	2024-03-12 16:58:42
Merge branch 'develop' of https://github.com/AndryFodor/WindowsFileMan	Andry <andrijfodor@gmail.c	2024-03-04 10:24:07
Update README.md	Andry <122674916+AndryF	2024-02-26 15:26:37
fixed readme.md	Andry <andrijfodor@gmail.c	2024-03-04 10:22:54

SHA1 ID: 95046fa2df2c064f33885c30b4be09694507e7a3 Row 1 / 43

Find commit containing: Exact All

Puc. 15. Git development branches

## РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ

У цьому розділі буде описано поетапне створення програмного забезпечення.

### 3.1. Основні функції програми

У даному розділі буде представлена реалізація програмного рішення для файлового менеджера на основі коду (додаток). Наведений код містить наступні складові, які будуть розглянуті детальніше:

Функція `void clearScreen()`: Ця функція очищає екран консолі шляхом виклику системної команди "cls".

Функція `void printMenu(const vector<string>& options, const fs::path currentPath)`: Ця функція виводить на екран меню з переданого вектора options. Кожен пункт меню номерується і виводиться на окремому рядку. Також в кінці після меню виводиться інформація про програмну підтримку таких команд, як cd та ls, що є доволі зручними у використанні

Функція `void displayDirectoryContents(const fs::path& path)`: Ця функція відображає вміст директорії, вказаної шляхом path. Вона перебирає всі елементи в директорії та виводить їх на екран, показуючи тип (файл або директорія), назву кожного елемента та час останньої зміни. Ця ж функція відпрацьовує і при виконанні команди ls.

Функції `void deleteFile(const fs::path& path)` та `void deleteDirectory(const fs::path& path)`: Ці функції видаляють вказаний файл або директорію. Вони перевіряють, чи існує файл або директорія за вказаним шляхом, і видаляють його за допомогою функції remove з бібліотеки filesystem.

Функції `void createDirectory(const fs::path& path)` та `void createFile(const fs::path& path)`: Ці функції створюють нову директорію або

файл за вказаним шляхом. Вони використовують функції `create_directory` або `ofstream` для створення директорії або файлу відповідно.

Функція `void displayFileProperties(const fs::path& path)`: Ця функція виводить властивості вказаного файлу. Вона перевіряє, чи існує файл за вказаним шляхом, і виводить його шлях, розмір та права доступу.

Функція `void openFile(const fs::path& filePath)`: Ця функція відкриває вказаний файл та виводить його вміст на екран. Вона перевіряє, чи існує файл за вказаним шляхом, а потім використовує `ifstream` для відкриття файлу та виведення його вмісту на екран. Варто зауважити, що ця функція коректно виводить вміст текстових файлів (txt, js, py, css, html), тоді як відкрити файли відеоформату та вивести коректно їх вміст вона не зможе.

Функція `void renameFile(const fs::path& filePath, const string& newFileName)` та `void renameDirectory(const fs::path& dirPath, const string& newDirName)`: Ці функції перевіряють правильність шляху та перейменовують відповідно файл або дерикторію. В результаті успішного завершення операції користувач може побачити про це вивів та перевірити командою `ls` або відповідною з меню те, що файл чи дерикторія і справді змінилася.

Функція `void copyDirectory(const fs::path& sourceDirPath, const fs::path& destinationDirPath)` та `void copyFile(const fs::path& sourceFilePath, const fs::path& destinationFilePath)` : Відповідні функції використовують для виявлення помилок в шляху чи щось інше за допомогою блока `try catch`. Це забезпечує докладнішу інформацію у випадку виникнення помилки при копіюванні дерикторій та файлів. Результатом виконання першої функції буде скопійована дерикторія із усім вмістом на новому шляху, який примається другим параметром. Аналогічно відьбувається з файлом. Також важливий момент полягає в тому, що копіювати до місця призначення можна файл чи дерикторію і під іншим



іменем, що поєднує в собі відразу функціонал копіювання та перейменування, що є доволі зручною функцією.

Функція `moveDirectory(const fs::path& sourceDirPath, const fs::path& destinationDirPath)` та `moveFile(const fs::path& sourceFilePath, const fs::path& destinationFilePath)`: Відповідні функції також виконують необхідні операції в блоці `try catch`. Працюють вони дуже схожим чином до попередніх двох, але в результаті дерикторія та файл будуть не скопійовані за новим шляхом, а переміщені за новим шляхом, а за старим вже не можна буде їх знайти.

Функція `void absolutePath(fs::path& enteredPath, const fs::path& currentPath)`: Важлива функція для коректної роботи всієї утиліти. Ця функція забезпечує отримання абсолютного шляху на основі введеного та поточного. Ця функція в поєднанні з іншим функціоналом підтримує можливість використання в утиліті відносних шляхів, що є дуже корисною та зручною функцією для користувача. Він може ввести як відносний шлях до файлу чи дерикторії, так і відносний. Файловий менеджер автоматично визначить, чи це абсолютний шлях, чи відносний. Якщо відносний – він буде перетворений в абсолютний перед тим, як потрапити до відповідної функції обробки.

Головна функція `main()`: Ця функція є точкою входу в програму. Вона виводить вітання до користувача та створює основне меню, де користувач може вибрати різні опції, такі як перегляд дерикторії, створення або видалення файлів/дерикторій, перегляд властивостей файлу, відкриття файлу.

Програмне рішення надає користувачу зручний консольний інтерфейс для взаємодії з файловою системою. Користувач може легко переглядати вміст дерикторій, створювати та видаляти файли та дерикторії, переглядати властивості файлів і відкривати файли дляперегляду їх вмісту. Це дозволяє ефективно керувати файловою системою без необхідності використання командного рядка або інших засобів.

Загальна структура програмного рішення базується на використанні бібліотеки `filesystem`, яка надає потужні інструменти для роботи з файловою системою в мові програмування C++. Кожна функція відповідає за певну операцію, яка може бути виконана з файловою системою, і забезпечує її виконання з використанням відповідних функцій бібліотеки `filesystem`.

Таке програмне рішення може бути використане в різних сценаріях, де потрібно працювати з файловою системою, таких як створення, перегляд і видалення файлів та директорій, а також отримання властивостей файлів. Воно може бути корисним як для індивідуальних користувачів, так і для програмістів, які розробляють програми, що працюють з файлами і директоріями.

## РОЗДІЛ 4. ТЕСТУВАННЯ

### 4.1. Інструкція для користувача

Запустіть програму "File Explorer". Побачивши вітання, натисніть "Enter".

Перед вами з'явиться список доступних опцій:

- Вийти з програми
- Вивести зміст поточного каталогу
- Змінити поточний каталог
- Створити файл
- Створити каталог
- Видалити файл
- Видалити каталог
- Вивести властивості файлу
- Відкрити файл
- перейменувати файл
- перейменувати каталог
- Скопіювати файл
- Скопіювати каталог
- Перемістити файл
- Перемістити каталог

Ви можете вибрати опцію, натиснувши відповідну цифру.

Залежно від обраної опції, програма може попросити ввести додаткові дані, наприклад, шлях до файлу або каталогу.

Після виконання операції програма може вивести повідомлення з про успішне виконання операції або повідомлення про помилку.

Натисніть будь-яку клавішу, щоб продовжити після виконання операції.

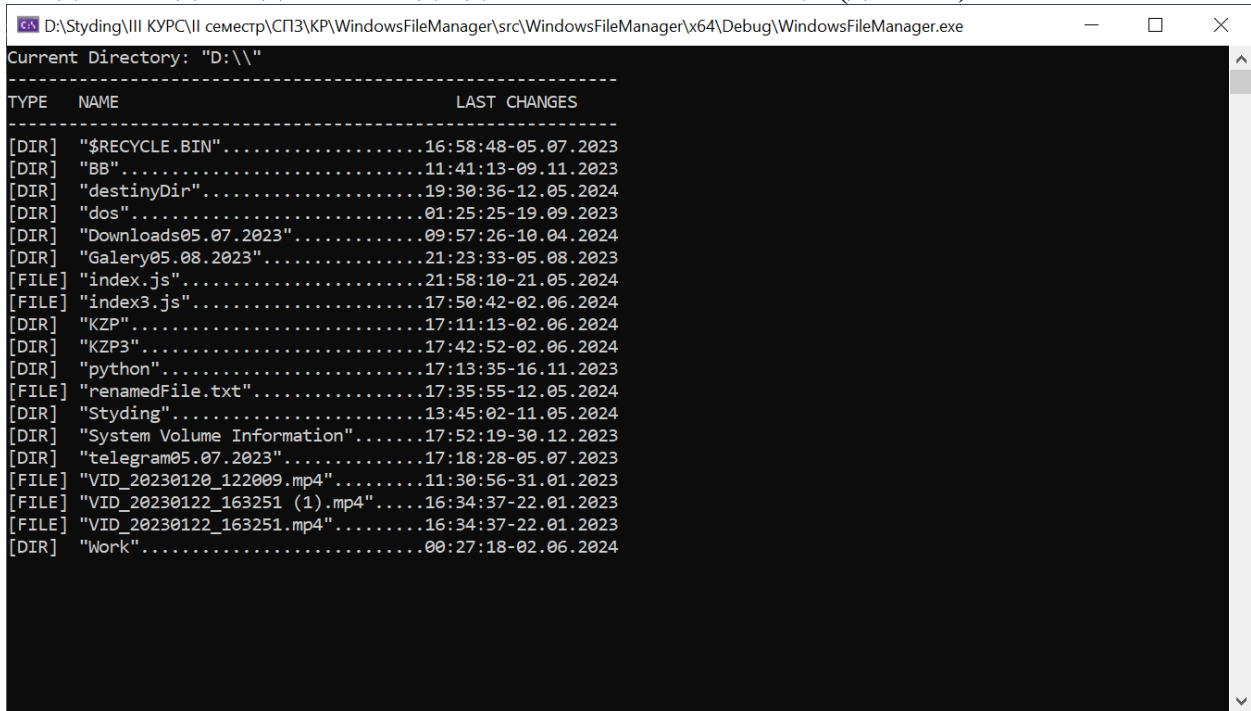
Продовжуйте вибирати опції або натисніть "1" для виходу з програми.

## 4.2. Тестування програми

В цьому розділі проведений детальний опис дій, які виконуються для тестування та знаходження дефектів в системній утиліті WindowsFileManager.

### 1. Тестування команди 2 та ls

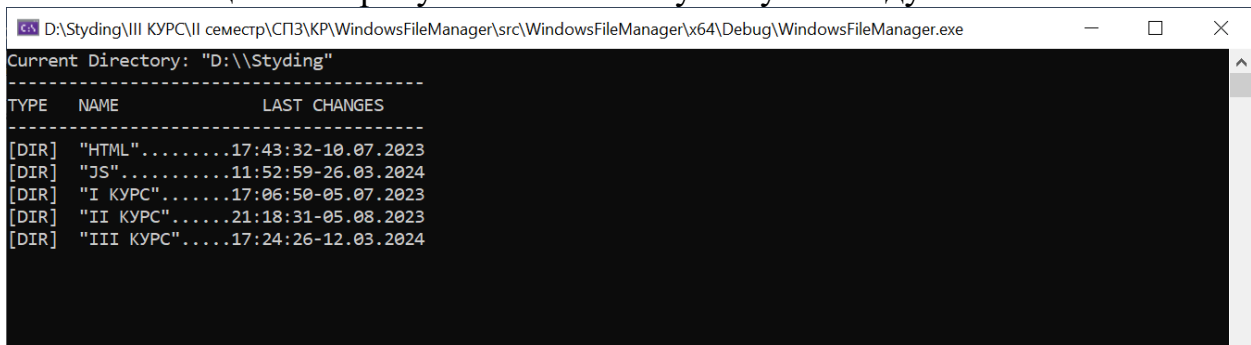
Вводимо відповідні команди для початкової локації (диск D)



```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Current Directory: "D:\\"
-----
TYPE      NAME                                           LAST CHANGES
-----
[DIR]     "$RECYCLE.BIN".....16:58:48-05.07.2023
[DIR]     "BB".....11:41:13-09.11.2023
[DIR]     "destinyDir".....19:30:36-12.05.2024
[DIR]     "dos".....01:25:25-19.09.2023
[DIR]     "Downloads05.07.2023".....09:57:26-10.04.2024
[DIR]     "Galery05.08.2023".....21:23:33-05.08.2023
[FILE]    "index.js".....21:58:10-21.05.2024
[FILE]    "index3.js".....17:50:42-02.06.2024
[DIR]     "KZP".....17:11:13-02.06.2024
[DIR]     "KZP3".....17:42:52-02.06.2024
[DIR]     "python".....17:13:35-16.11.2023
[FILE]    "renamedFile.txt".....17:35:55-12.05.2024
[DIR]     "Styding".....13:45:02-11.05.2024
[DIR]     "System Volume Information".....17:52:19-30.12.2023
[DIR]     "telegram05.07.2023".....17:18:28-05.07.2023
[FILE]    "VID_20230120_122009.mp4".....11:30:56-31.01.2023
[FILE]    "VID_20230122_163251 (1).mp4".....16:34:37-22.01.2023
[FILE]    "VID_20230122_163251.mp4".....16:34:37-22.01.2023
[DIR]     "Work".....00:27:18-02.06.2024
```

Рис. 16. Тестування команди 2 та ls, крок 1

Змінімо локацію та спробуємо виконати ту саму команду



```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Current Directory: "D:\Styding"
-----
TYPE      NAME                                           LAST CHANGES
-----
[DIR]     "HTML".....17:43:32-10.07.2023
[DIR]     "JS".....11:52:59-26.03.2024
[DIR]     "I КУРС".....17:06:50-05.07.2023
[DIR]     "II КУРС".....21:18:31-05.08.2023
[DIR]     "III КУРС".....17:24:26-12.03.2024
```

Рис. 17. Тестування команди 2 та ls, крок 2

Перемістимося в пусту дерикторію та спробуємо виконати цю операцію там

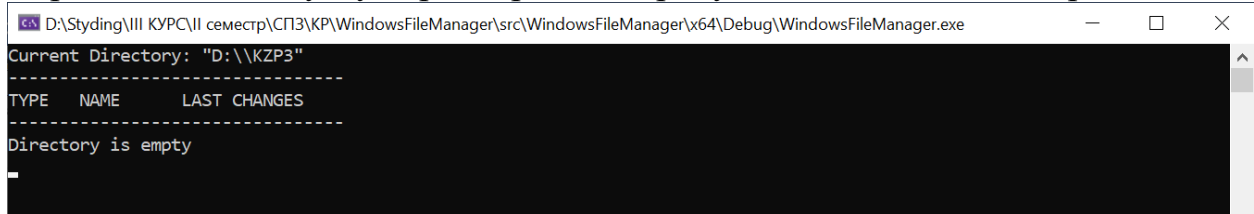


Рис. 18. Тестування команди 2 та ls, крок 3

**Висновок: команда 2 та команда ls працюють коректно та не видають жодних можливих дефектів у всіх можливих ситуаціях**

## 2. Тестування команди 3 та cd

Для команди 3, яка очікує абсолютний шлях, введемо відносний і подивимося результат

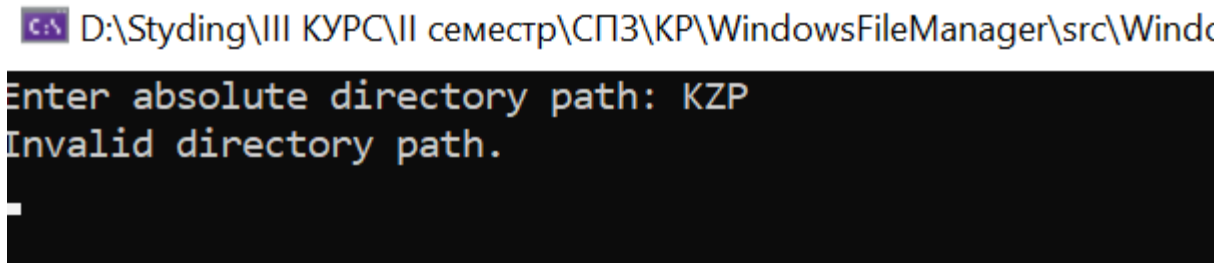


Рис. 19. Тестування команди 3 та cd, крок 1

Введемо цей самий відносний шлях для команди cd

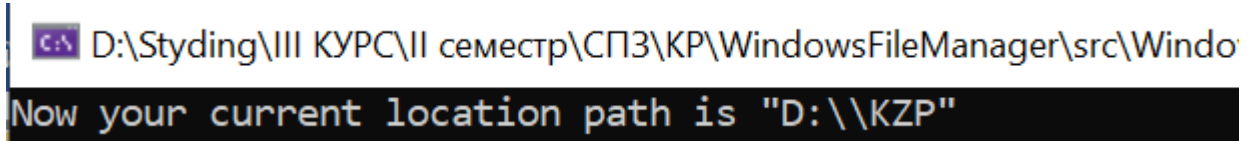


Рис. 20. Тестування команди 3 та cd, крок 2

Введемо специфічний шлях «..», що означає вийти на верхній рівень

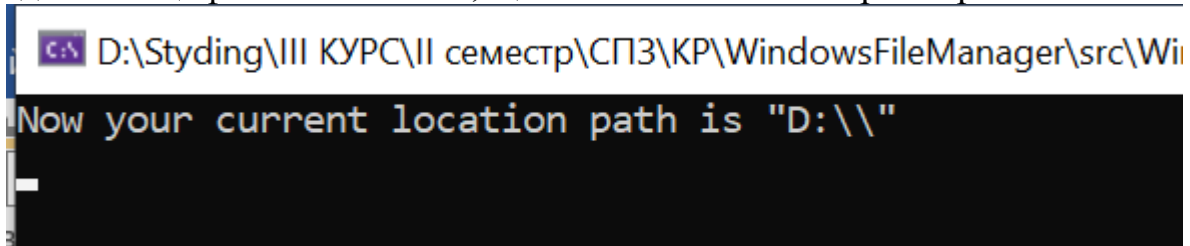
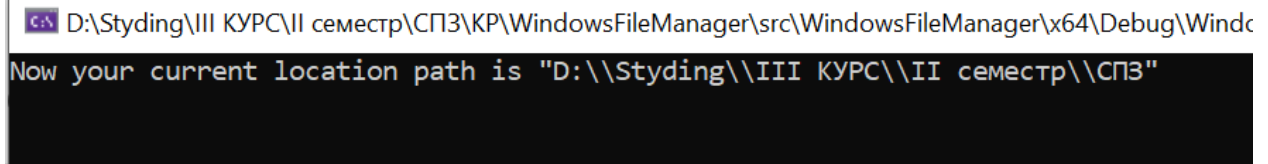


Рис. 21. Тестування команди 3 та cd, крок 3

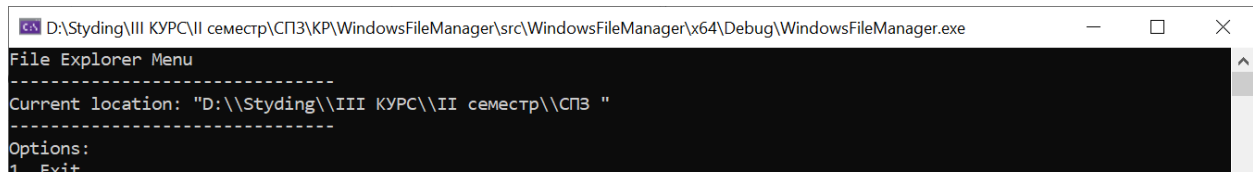
Введемо шлях D:\Styding\III КУРС\II семестр\СПЗ для команди cd



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\KP\WindowsFileManager\src\WindowsFileManager\x64\Debug\Windc
Now your current location path is "D:\\Styding\\III КУРС\\II семестр\\СПЗ"
```

*Рис. 22. Тестування команди 3 та cd, крок 4*

Введемо цей же шлях для команди 3



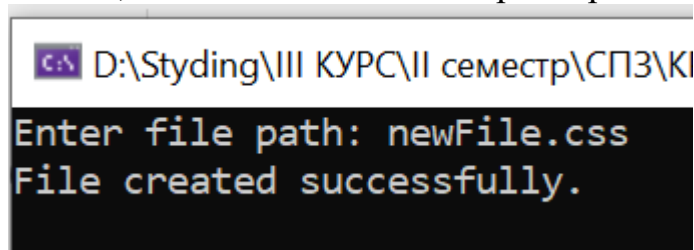
```
D:\Styding\III КУРС\II семестр\СПЗ\KP\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
File Explorer Menu
-----
Current location: "D:\\Styding\\III КУРС\\II семестр\\СПЗ "
-----
Options:
1 Exit
```

*Рис. 23. Тестування команди 3 та cd, крок 5*

**Висновок: команда 3 та команда cd працюють коректно та не видають жодних можливих дефектів у всіх можливих ситуаціях**

### 3. Тестування команди 4

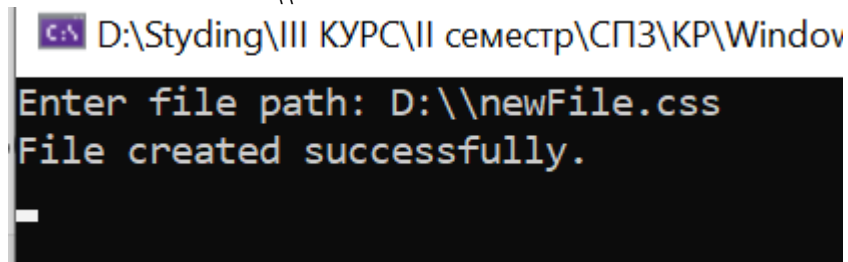
Введемо відносний шлях, коли знаходимося в дерикторії KZP3 newFile.css



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\KI
Enter file path: newFile.css
File created successfully.
```

*Рис. 24. Тестування команди 4, крок 1*

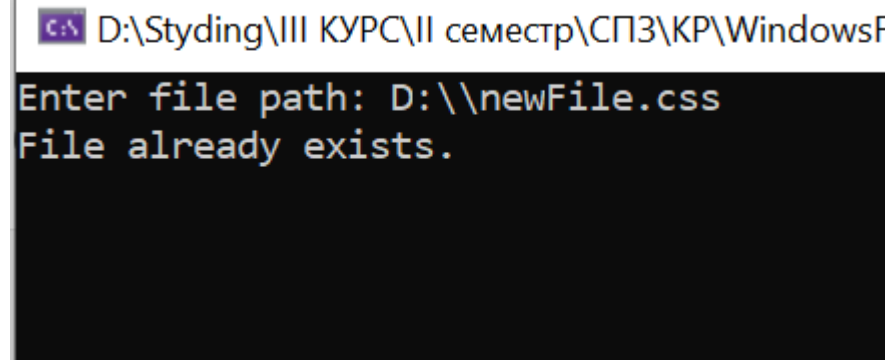
Введемо абсолютний шлях D:\\newFile.css



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\KP\Window
Enter file path: D:\\newFile.css
File created successfully.
```

*Рис. 25. Тестування команди 4, крок 2*

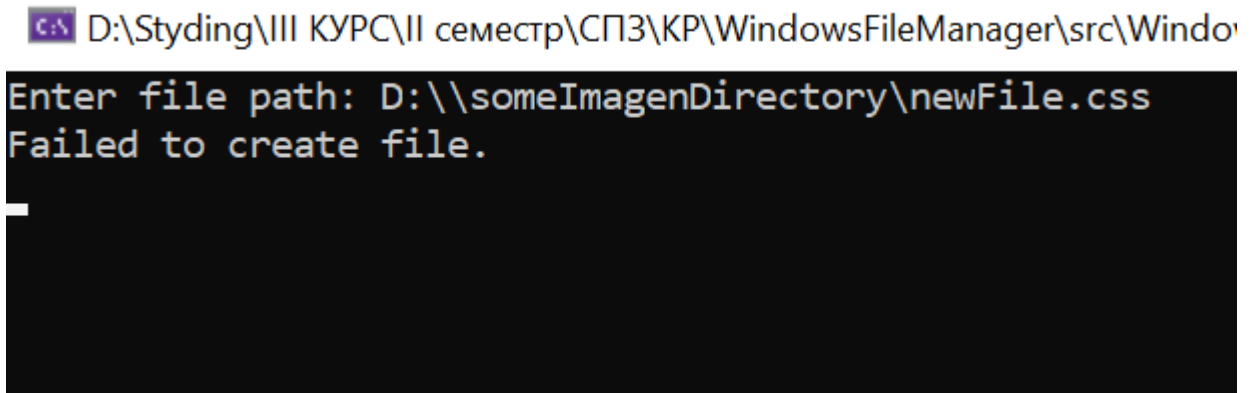
Введемо цей же шлях

A screenshot of a Windows command prompt window. The title bar shows the path 'D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsF'. The prompt is 'C:\>'. The user has entered 'D:\\\newFile.css'. The command prompt displays the message 'Enter file path: D:\\newFile.css' followed by 'File already exists.' on the next line.

```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsF
Enter file path: D:\\newFile.css
File already exists.
```

*Рис. 26. Тестування команди 4, крок 3*

Введемо некоректний абсолютний шлях для створення файлу

A screenshot of a Windows command prompt window. The title bar shows the path 'D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\Windo'. The prompt is 'C:\>'. The user has entered 'D:\\someImagenDirectory\\newFile.css'. The command prompt displays the message 'Enter file path: D:\\someImagenDirectory\\newFile.css' followed by 'Failed to create file.' on the next line.

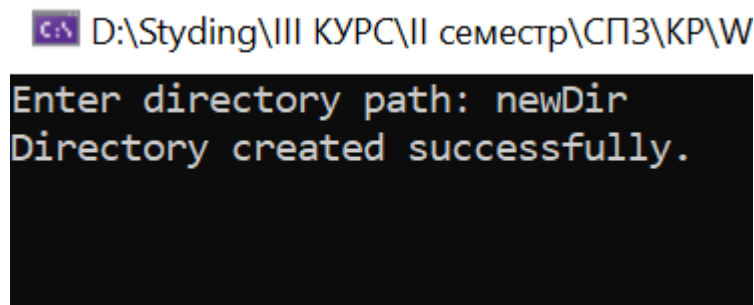
```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\Windo
Enter file path: D:\\someImagenDirectory\\newFile.css
Failed to create file.
```

*Рис. 27. Тестування команди 4, крок 4*

**Висновок:** команда 4 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях

#### 4. Тестування команди 5

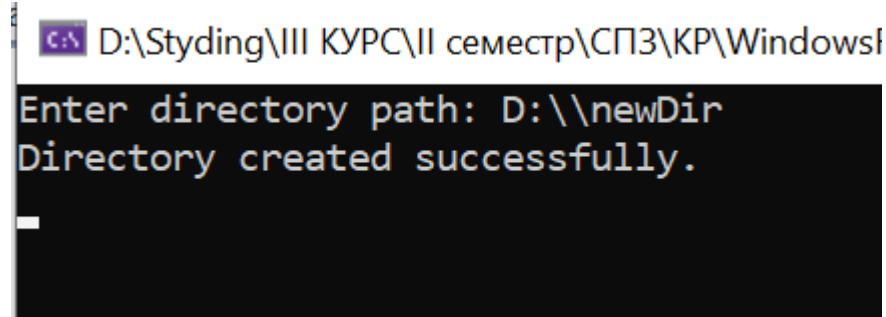
Знаходячись в дерикторії KZP3, введемо відносний шлях до нової дерикторії newDir

A screenshot of a Windows command prompt window. The title bar shows the path 'D:\Styding\III КУРС\II семестр\СПЗ\КР\W'. The prompt is 'C:\>'. The user has entered 'newDir'. The command prompt displays the message 'Enter directory path: newDir' followed by 'Directory created successfully.' on the next line.

```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\W
Enter directory path: newDir
Directory created successfully.
```

*Рис. 28. Тестування команди 5, крок 1*

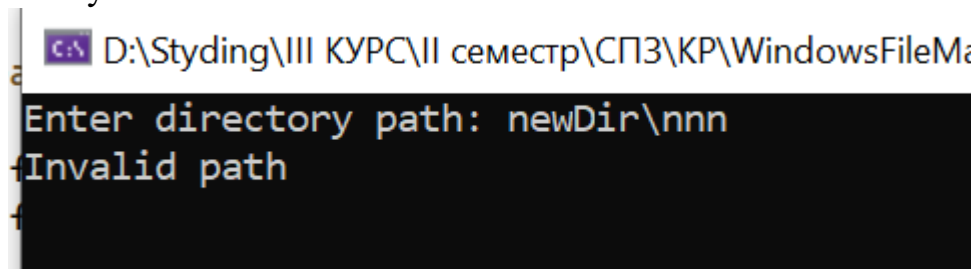
Введемо абсолютний шлях



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsI  
Enter directory path: D:\\newDir  
Directory created successfully.  
_
```

*Рис. 29. Тестування команди 5, крок 2*

Введемо неіснуючий шлях



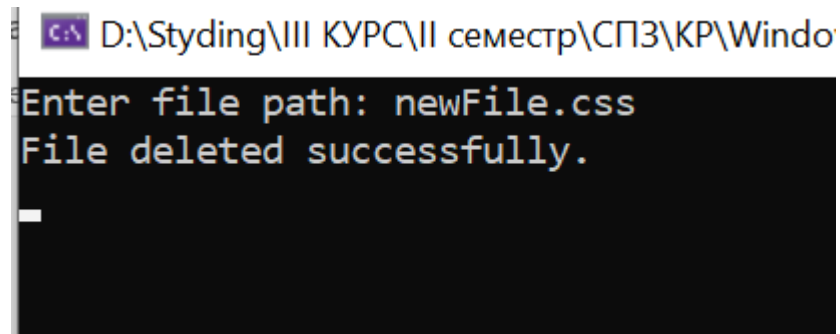
```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileMa  
Enter directory path: newDir\nnn  
Invalid path  
_
```

*Рис. 30. Тестування команди 5, крок 3*

**Висновок:** команда 5 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях

#### 5. Тестування команди 6

Знаходячись в дерикторії KZP3, в якій щойно був створений файл newFile.css, введемо відносний шлях newFile.css

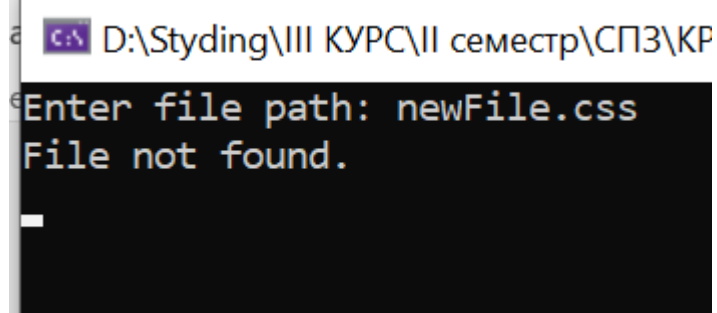


```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\Windo  
Enter file path: newFile.css  
File deleted successfully.  
_
```

*Рис. 31. Тестування команди 6, крок 1*



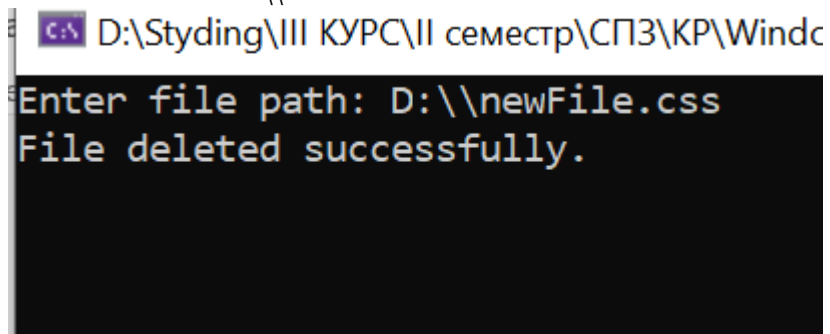
Виконаємо цю операцію ще раз



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР
Enter file path: newFile.css
File not found.
```

*Рис. 32. Тестування команди 6, крок 2*

Введемо абсолютний шлях D:\\newFile.css



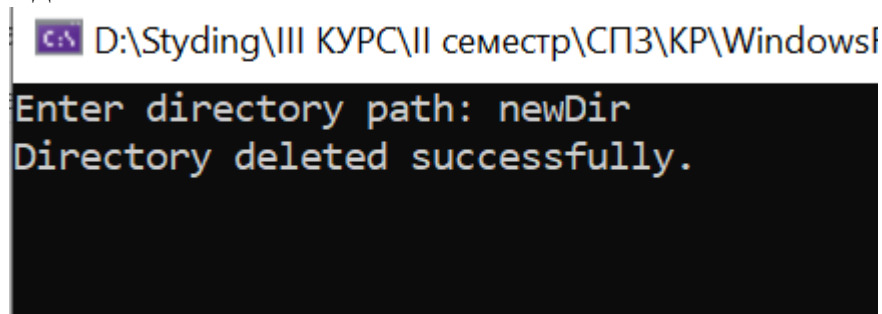
```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\Windc
Enter file path: D:\\newFile.css
File deleted successfully.
```

*Рис. 33. Тестування команди 6, крок 3*

**Висновок:** команда 6 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях

#### 6. Тестування команди 7

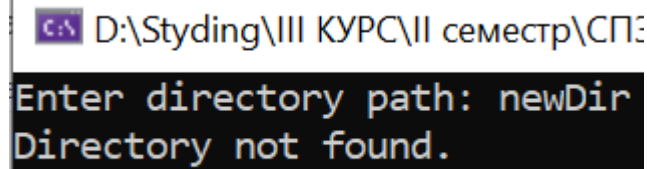
Знаходячись в дерикторії KZP3, в якій щойно була створена дерикторія newDir, введемо відносний шлях newDir



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsI
Enter directory path: newDir
Directory deleted successfully.
```

*Рис. 34. Тестування команди 7, крок 1*

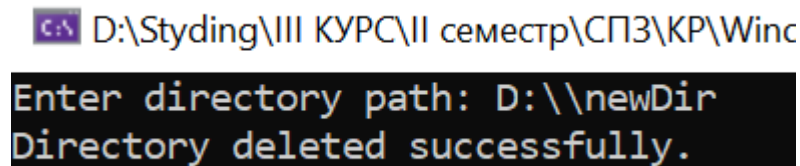
Виконаємо цю операцію ще раз



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ
Enter directory path: newDir
Directory not found.
```

Рис. 35. Тестування команди 7, крок 2

Введемо абсолютний шлях D:\\newDir



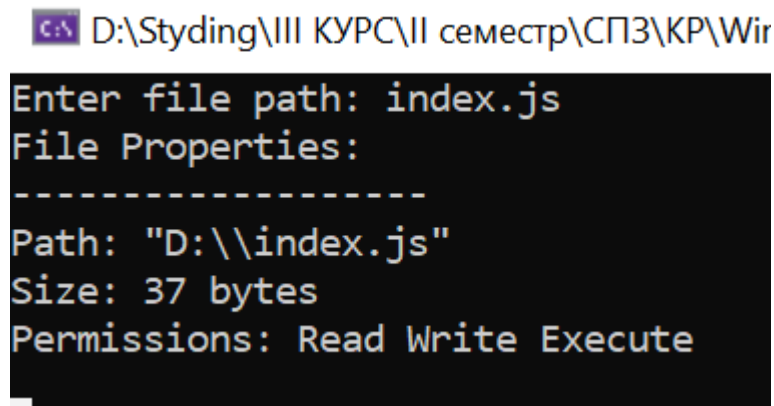
```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\Winс
Enter directory path: D:\\newDir
Directory deleted successfully.
```

Рис. 36. Тестування команди 7, крок 3

**Висновок: команда 7 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях**

#### 7. Тестування команди 8

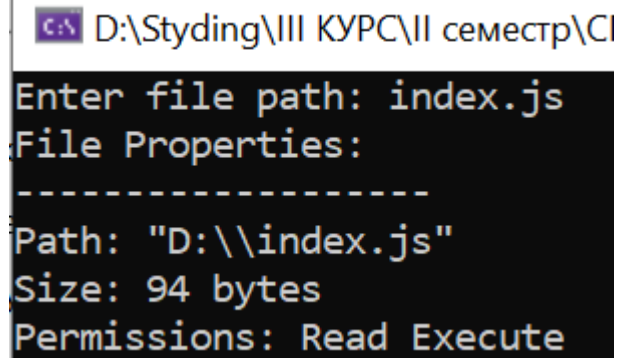
Знаходячись за локацією “D:\\”, введу index.js



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\Wir
Enter file path: index.js
File Properties:
-----
Path: "D:\\index.js"
Size: 37 bytes
Permissions: Read Write Execute
```

Рис. 37. Тестування команди 8, крок 1

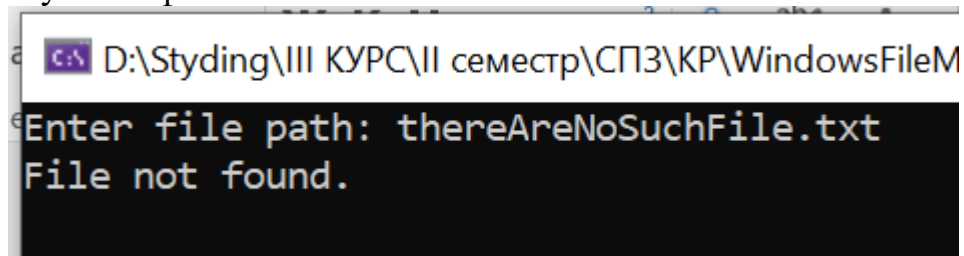
Зміню атрибути файлу та його вміст



```
C:\N D:\Styding\III КУРС\II семестр\CI
Enter file path: index.js
File Properties:
-----
Path: "D:\\index.js"
Size: 94 bytes
Permissions: Read Execute
```

Рис. 38. Тестування команди 8, крок 2

Введу неіснуючий файл



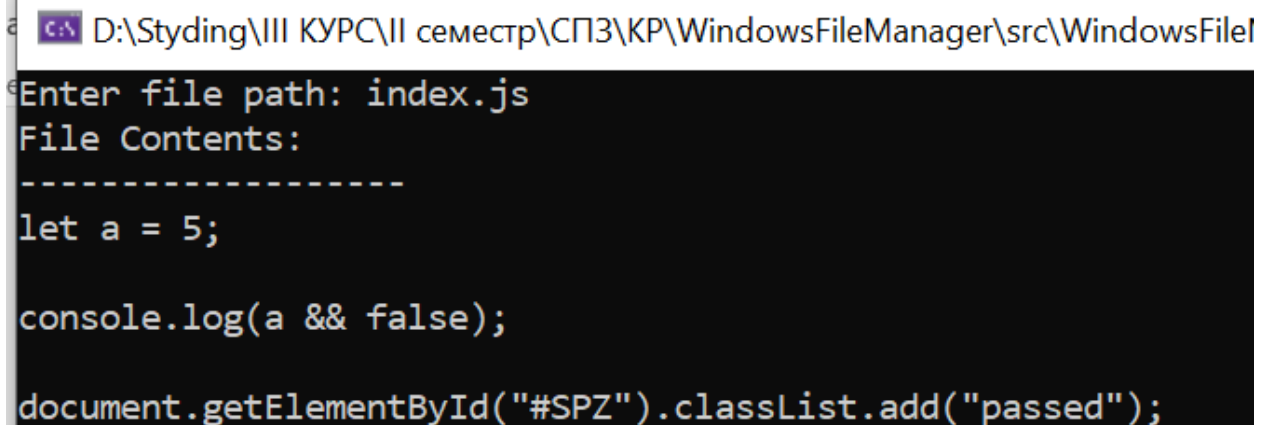
```
C:\N D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileM
Enter file path: thereAreNoSuchFile.txt
File not found.
```

Рис. 39. Тестування команди 8, крок 3

**Висновок:** команда 8 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях

#### 8. Тестування команди 9

Знаходячись за локацією “D:\\”, введу index.js



```
C:\N D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileI
Enter file path: index.js
File Contents:
-----
let a = 5;

console.log(a && false);

document.getElementById("#SPZ").classList.add("passed");
```

Рис. 40. Тестування команди 9, крок 1

Також спробуємо відкрити файл renamedFile.txt

```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileM.  
Enter file path: renamedFile.txt  
File Contents:  
-----  
skjdnfjksdnfjsdnfkjsdn fkjsdnkfj nsdkjfsdkjnf skjdnf kjsdnf
```

Рис. 41. Тестування команди 9, крок 2

Спробуємо відкрити формат mp4

```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe  
Enter file path: VID_20230120_122009.mp4  
File Contents:  
-----  
ftypmp42isommp42-=-free1mvhdЯр±Яр±'±, ,b±±±±±@±vmeta!hd1rmdta+keys±dtacom.android.version"ilst
```

Рис. 42. Тестування команди 9, крок 3

Введемо неіснуючий файл

```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\  
Enter file path: sldjcnsd.cs  
Invalid file path.
```

Рис. 43. Тестування команди 9, крок 4

**Висновок:** команда 9 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях. Проте передбачена можливість отримання контексту лише файлів з текстовим контентом

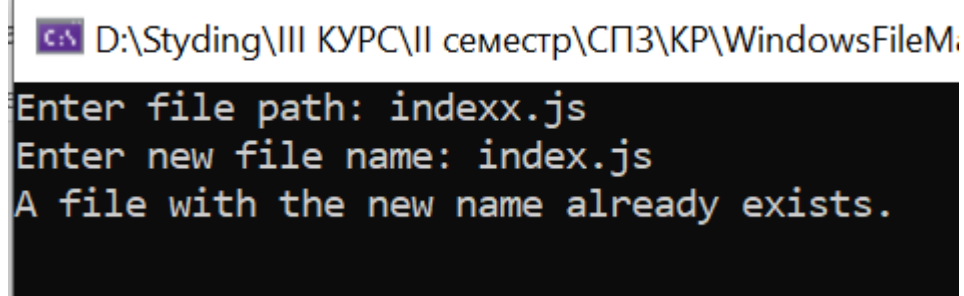
## 9. Тестування команди 10

Знаходячись за локацією “D:\”, введемо відносний шлях до файлу newIndex.js та нове ім’я.

```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\Wir  
Enter file path: newIndex.js  
Enter new file name: indexX.js  
File renamed successfully.
```

Рис. 44. Тестування команди 10, крок 1

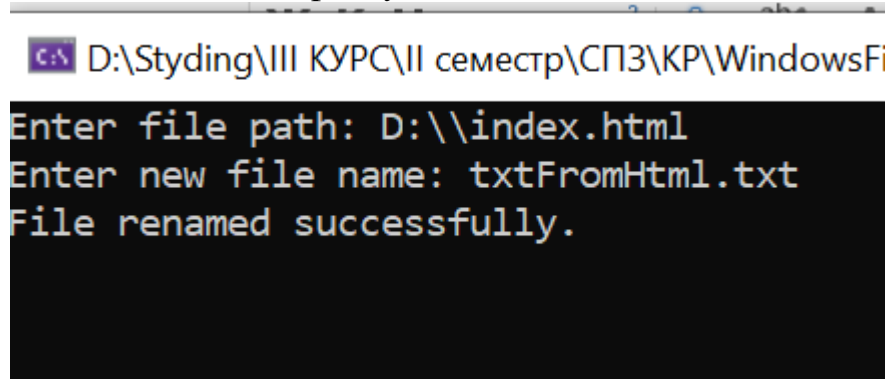
Спробуємо перейменувати файл на вже існуючий



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileM
Enter file path: indexx.js
Enter new file name: index.js
A file with the new name already exists.
```

Рис. 45. Тестування команди 10, крок 2

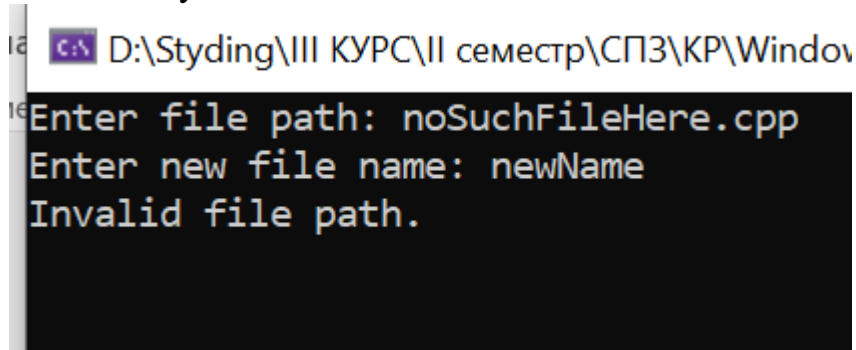
Введемо абсолютний шлях до файлу



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFi
Enter file path: D:\\index.html
Enter new file name: txtFromHtml.txt
File renamed successfully.
```

Рис. 46. Тестування команди 10, крок 3

І нарешті введемо неіснуючий шлях



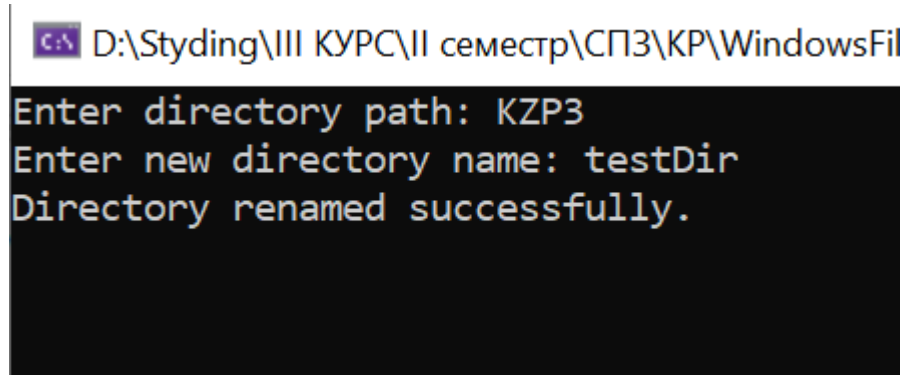
```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\Window
Enter file path: noSuchFileHere.cpp
Enter new file name: newName
Invalid file path.
```

Рис. 47. Тестування команди 10, крок 4

**Висновок:** команда 10 працює коректно та не видає жодинх можливих дефектів у всіх можливих ситуаціях.

## 10. Тестування команди 11

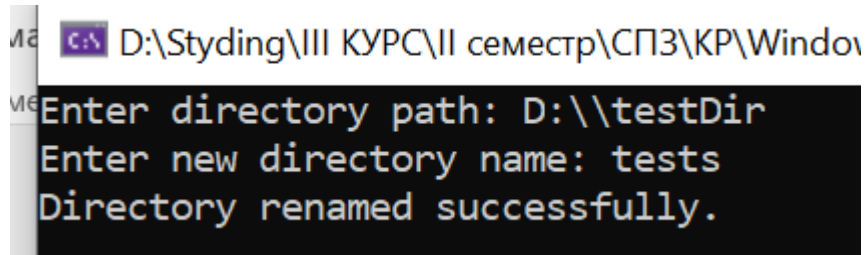
Знаходячись в “D:\”, введемо відносний шлях KZP3 та ім’я нової дерикторії як testDir



```
C:\N D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFil
Enter directory path: KZP3
Enter new directory name: testDir
Directory renamed successfully.
```

Рис. 48. Тестування команди 11, крок 1

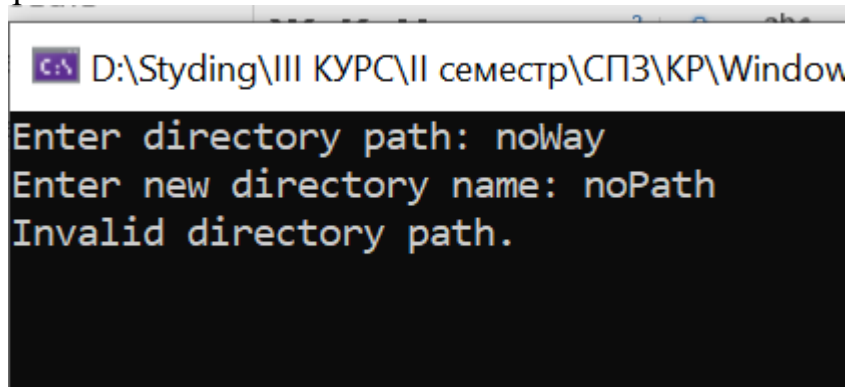
Введемо абсолютні шляхи



```
C:\N D:\Styding\III КУРС\II семестр\СПЗ\КР\Window
Enter directory path: D:\\testDir
Enter new directory name: tests
Directory renamed successfully.
```

Рис. 49. Тестування команди 11, крок 2

Введемо некоректний шлях



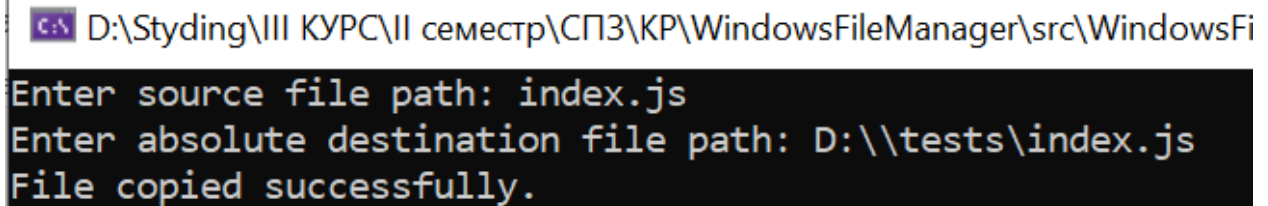
```
C:\N D:\Styding\III КУРС\II семестр\СПЗ\КР\Window
Enter directory path: noWay
Enter new directory name: noPath
Invalid directory path.
```

Рис. 50. Тестування команди 11, крок 3

**Висновок:** команда 11 працює коректно та не видає жодинх можливих дефектів у всіх можливих ситуаціях.

## 11. Тестування команди 12

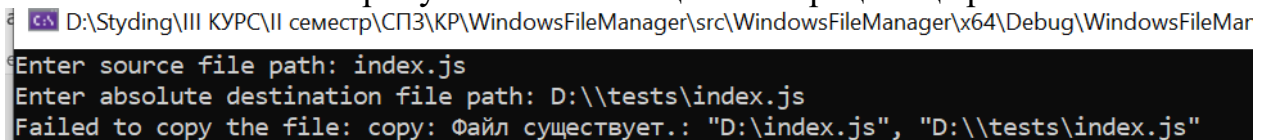
Як відомо, для виконання цієї операції необхідно вводити в будь-якому випадку абсолютний шлях до місця, куди треба скопіювати файл + його назву (не обов'язково поточну). Знаходячись в "D:\\", введемо відносний шлях index.js до файлу та D:\\tests\\index.js як шлях, куди треба спопіювати файл



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFi
Enter source file path: index.js
Enter absolute destination file path: D:\\tests\\index.js
File copied successfully.
```

Рис. 51. Тестування команди 12, крок 1

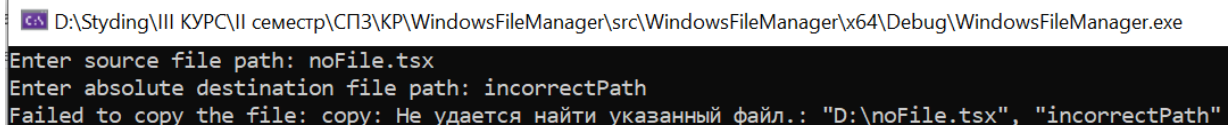
Спробуємо виконати цю ж операцію ще раз



```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileMar
Enter source file path: index.js
Enter absolute destination file path: D:\\tests\\index.js
Failed to copy the file: copy: Файл существует.: "D:\\index.js", "D:\\tests\\index.js"
```

Рис. 52. Тестування команди 12, крок 2

Спробуємо вказати некоректний шлях



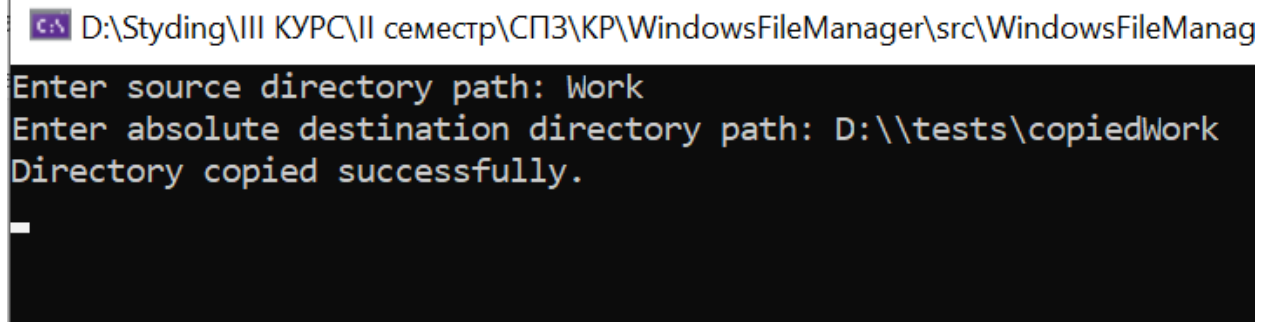
```
C:\> D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source file path: noFile.tsx
Enter absolute destination file path: incorrectPath
Failed to copy the file: copy: Не удастся найти указанный файл.: "D:\\noFile.tsx", "incorrectPath"
```

Рис. 53. Тестування команди 12, крок 3

**Висновок:** команда 12 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях.

## 12. Тестування команди 13

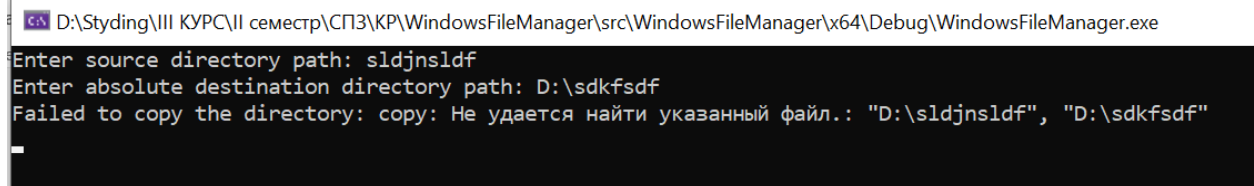
Перебуваючи на диску D, введемо відносний шлях до дерикторії та абсолютний з новим іменем вкінці шляху



```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManag
Enter source directory path: Work
Enter absolute destination directory path: D:\\tests\\copiedWork
Directory copied successfully.
```

Рис. 54. Тестування команди 13, крок 1

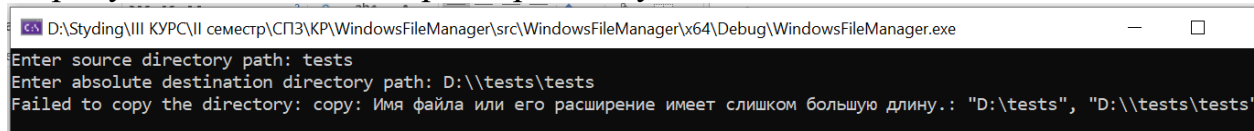
Введемо некоректні шляхи



```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source directory path: sldjnsldf
Enter absolute destination directory path: D:\sdkfsdf
Failed to copy the directory: copy: Не удается найти указанный файл.: "D:\sldjnsldf", "D:\sdkfsdf"
```

Рис. 55. Тестування команди 13, крок 2

Спробуємо скопіювати дерикторію саму в себе



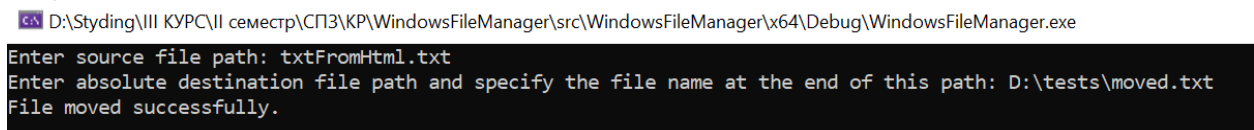
```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source directory path: tests
Enter absolute destination directory path: D:\\tests\\tests
Failed to copy the directory: copy: Имя файла или его расширение имеет слишком большую длину.: "D:\tests", "D:\\tests\\tests"
```

Рис. 56. Тестування команди 13, крок 3

**Висновок: команда 13 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях.**

## 13. Тестування команди 14

З диску D перенесемо файл txtFromHtml.txt за таким шляхом і новим іменем D:\tests\moded.txt

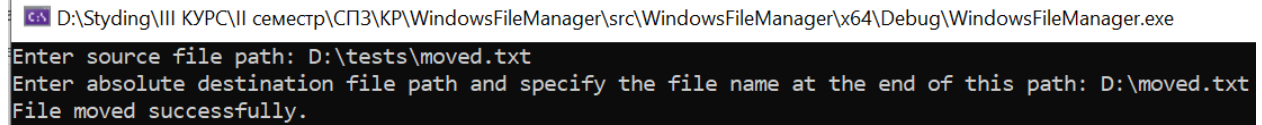


```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source file path: txtFromHtml.txt
Enter absolute destination file path and specify the file name at the end of this path: D:\tests\moved.txt
File moved successfully.
```

Рис. 57. Тестування команди 14, крок 1



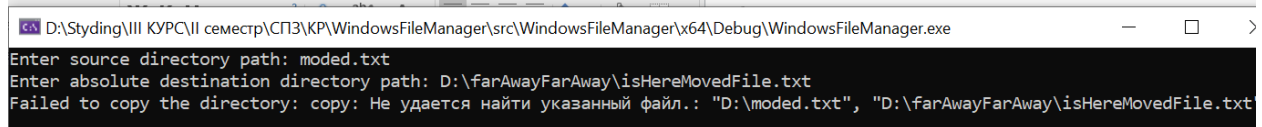
Перевіримо на коректність опрацювання абсолютних шляхів



```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source file path: D:\tests\moved.txt
Enter absolute destination file path and specify the file name at the end of this path: D:\moved.txt
File moved successfully.
```

*Рис. 58. Тестування команди 14, крок 2*

Введемо некоректні шляхи



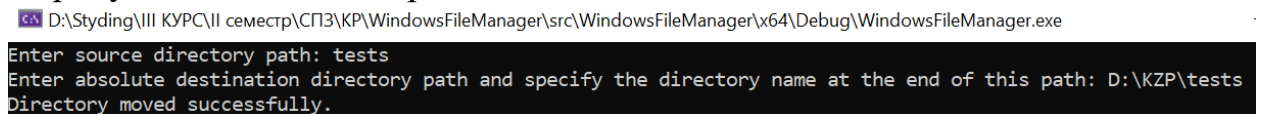
```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source directory path: moded.txt
Enter absolute destination directory path: D:\farAwayFarAway\isHereMovedFile.txt
Failed to copy the directory: copy: Не удастся найти указанный файл.: "D:\moded.txt", "D:\farAwayFarAway\isHereMovedFile.txt"
```

*Рис. 59. Тестування команди 14, крок 3*

**Висновок:** команда 14 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях.

#### 14. Тестування команди 15

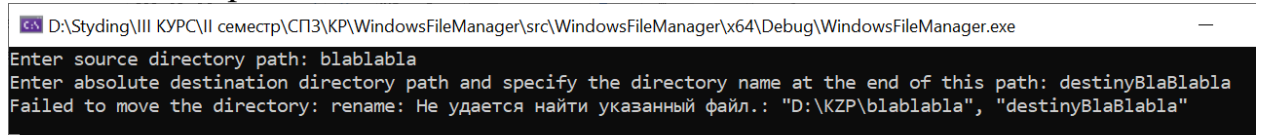
Спробуємо виконати операцію



```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source directory path: tests
Enter absolute destination directory path and specify the directory name at the end of this path: D:\KZP\tests
Directory moved successfully.
```

*Рис. 60. Тестування команди 15, крок 1*

Введемо некоректні шляхи



```
D:\Styding\III КУРС\II семестр\СПЗ\КР\WindowsFileManager\src\WindowsFileManager\x64\Debug\WindowsFileManager.exe
Enter source directory path: blablalba
Enter absolute destination directory path and specify the directory name at the end of this path: destinyBlaBlaBla
Failed to move the directory: rename: Не удастся найти указанный файл.: "D:\KZP\blablalba", "destinyBlaBlaBla"
```

*Рис. 61. Тестування команди 15, крок 2*

**Висновок:** команда 15 працює коректно та не видає жодних можливих дефектів у всіх можливих ситуаціях.

#### **4.3. Загальний висновок тестування програми**

В результаті ретельного тестування системної утиліти WindowsFileManager було виявлено один дефект, але він був негайно виправлений. Він заключався в тому, що при створенні дерикторії не перевірялося, чи існує шлях, за яким її намагаються створити. Точніше, слід було перевіряти батьківський шлях, де повинна була знаходитися нова дерикторія. Відшукану помилку було успішно виправлено.

Більше дефектів знайдено не було. Програма готова до демонстрації

## ВИСНОВКИ

В результаті виконання курсового проекту було розроблено файлового провідника за допомогою мови програмування C++. Файловий менеджер є системною утилітою, що надає користувачеві можливість взаємодіяти з файловою системою операційної системи, здійснювати різноманітні операції над файлами та каталогами.

У роботі були використані основні бібліотеки C++, такі як `<iostream>`, `<filesystem>`, `<fstream>`, `<string>`, `<vector>` і `<conio.h>`. Бібліотека `<filesystem>` надає зручний інтерфейс для роботи з файловою системою, дозволяючи виконувати операції створення, видалення, перейменування, копіювання та переміщення файлів та каталогів. Функціональність файлового провідника включає такі операції, як відображення змісту поточного каталогу, зміна поточного каталогу, створення файлів та каталогів, видалення файлів та каталогів, відображення властивостей файлів, відкриття файлів для перегляду, перейменування файлів та каталогів, копіювання та переміщення файлів та каталогів. Також ця утиліта підтримує відносні шляхи, що було реалізовано за допомогою відповідних перевірок після кожного введення шляху та функції, яка формує абсолютний шлях, аби передати його у відповідну функцію.

Користувацький інтерфейс реалізований за допомогою текстового виводу та введення з клавіатури. Користувач має можливість обирати різні опції з меню, вводити необхідні дані (наприклад, шлях до файлу або каталогу) і отримувати відповідні результати операцій.

В цілому, розроблений файловий провідник є зручним інструментом для управління файловою системою, він надає користувачу зручний консольний інтерфейс для виконання різноманітних операцій з файлами та каталогами.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - "Operating System Concepts" (Загальне розуміння операційних систем та файлових систем)
2. Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau - "Operating Systems: ThreeEasy Pieces" (Для поглибленого вивчення операційних систем)
3. Bill Blunden - "Software Exorcism: A Handbook for Debugging and Optimizing Legacy Code" (Для вивчення методів відлагодження та оптимізації коду)
4. Jeffrey Richter - "Windows via C/C++" (Розробка програмного забезпечення, спрямованого на операційну систему Windows)
5. Mark Russinovich, David A. Solomon, Alex Ionescu - "Windows Internals"
6. MSDN - Мережа розробників Майкрософт  
<http://msdn.microsoft.com/>

## ДОДАТОК

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <filesystem>
#include <fstream>
#include <string>
#include <vector>
#include <conio.h>
#include <Windows.h>
#include <ctime>

using namespace std;
namespace fs = std::filesystem;

void clearScreen()
{
    system("cls");
}

void printMenu(const vector<string>& options, const fs::path currentPath)
{
    clearScreen();
    cout << "File Explorer Menu" << endl;
    cout << "-----" << endl;
    cout << "Current location: " << currentPath << endl;
    cout << "-----" << endl;
    cout << "Options:" << endl;
    for (size_t i = 0; i < options.size(); i++)
    {
        if (options.size() - i == 1)
            cout << options[i] << endl;
        else
            cout << i + 1 << ". " << options[i] << endl;
    }
}

void displayDirectoryContents(const fs::path& path)
{
    clearScreen();
    cout << "Current Directory: " << fs::absolute(path) << endl;

    int maxLength = 0;
    bool empty = true;
    for (const auto& entry : fs::directory_iterator(path)) {
        if (entry.path().filename().string().length() > maxLength) {
            maxLength = entry.path().filename().string().length();
        }
    }
    maxLength += 5;
    for (int i = 0; i < maxLength + 28; i++)
    {
        cout << "-";
    }
    cout << "\nTYPE    NAME";
    for (int i = 0; i < maxLength - 2; i++)
    {
        cout << " ";
    }
    cout << "    LAST CHANGES" << endl;
    for (int i = 0; i < maxLength + 28; i++)
    {
        cout << "-";
    }
}
```

```

cout << endl;
for (const auto& entry : fs::directory_iterator(path))
{
    if (fs::is_directory(entry.status()))
    {
        empty = false;
        cout << "[DIR] ";
    }
    else
    {
        empty = false;
        cout << "[FILE]";
    }

    cout << " " << entry.path().filename();

    auto file_time = fs::last_write_time(entry.path());

    auto time_point =
chrono::time_point_cast<chrono::system_clock::duration>(file_time -
fs::file_time_type::clock::now() + chrono::system_clock::now());

    time_t file_time_t = chrono::system_clock::to_time_t(time_point);

    struct tm* ptm = localtime(&file_time_t);
    char buffer[32];
    strftime(buffer, 32, "%H:%M:%S-%d.%m.%Y", ptm);

    for (int i = 0; i < maxLength - entry.path().filename().string().length(); i++)
    {
        cout << ".";
    }
    cout << buffer << endl;
}
if (empty)
    cout << "Directory is empty" << endl;
}

void deleteFile(const fs::path& path)
{
    if (fs::exists(path))
    {
        fs::remove(path);
        cout << "File deleted successfully." << endl;
    }
    else
    {
        cout << "File not found." << endl;
    }

    _getch();
}

void deleteDirectory(const fs::path& path)
{
    if (fs::exists(path) && fs::is_directory(path))
    {
        fs::remove_all(path);
        cout << "Directory deleted successfully." << endl;
    }
    else
    {
        cout << "Directory not found." << endl;
    }
}

```

```

        _getch();
    }

void createDirectory(const fs::path& path)
{
    if (fs::exists(path.parent_path())) {
        if (fs::create_directory(path))
        {
            cout << "Directory created successfully." << endl;
        }
        else
        {
            cout << "Failed to create directory." << endl;
        }
    }
    else {
        cout << "Invalid path" << endl;
    }

    _getch();
}

void createFile(const fs::path& path)
{
    if (fs::exists(path)) {
        cout << "File already exists." << endl;
        _getch();
        return;
    }

    ofstream file(path.string());
    if (file)
    {
        cout << "File created successfully." << endl;
    }
    else
    {
        cout << "Failed to create file." << endl;
    }
    file.close();

    _getch();
}

void displayFileProperties(const fs::path& path)
{
    if (fs::exists(path))
    {
        cout << "File Properties: " << endl;
        cout << "-----" << endl;
        cout << "Path: " << fs::absolute(path) << endl;
        cout << "Size: " << fs::file_size(path) << " bytes" << endl;

        cout << "Permissions: ";
        if ((fs::status(path).permissions() & fs::perms::owner_read) != fs::perms::none)
        {
            cout << "Read ";
        }
        if ((fs::status(path).permissions() & fs::perms::owner_write) != fs::perms::none)
        {
            cout << "Write ";
        }
        if ((fs::status(path).permissions() & fs::perms::owner_exec) != fs::perms::none)
        {

```

```

        cout << "Execute ";
    }
    cout << endl;
}
else
{
    cout << "File not found." << endl;
}

_getch();
}

void openFile(const fs::path& filePath)
{
    if (fs::exists(filePath) && fs::is_regular_file(filePath))
    {
        ifstream file(filePath);
        if (file.is_open())
        {
            cout << "File Contents: " << endl;
            cout << "-----" << endl;

            string line;
            while (getline(file, line))
            {
                cout << line << endl;
            }

            file.close();
        }
        else
        {
            cout << "Failed to open file." << endl;
        }
    }
    else
    {
        cout << "Invalid file path." << endl;
    }

    _getch();
}

void renameFile(const fs::path& filePath, const string& newFileName)
{
    if (fs::exists(filePath) && fs::is_regular_file(filePath))
    {
        fs::path parentPath = filePath.parent_path();
        fs::path newFilePath = parentPath / newFileName;

        if (fs::exists(newFilePath))
        {
            cout << "A file with the new name already exists." << endl;
        }
        else
        {
            try
            {
                fs::rename(filePath, newFilePath);
                cout << "File renamed successfully." << endl;
            }
            catch (const fs::filesystem_error& e)
            {
                cout << "Failed to rename file. Error: " << e.what() << endl;
            }
        }
    }
}

```



```

    }
}
else
{
    cout << "Invalid file path." << endl;
}
_getch();
}

void renameDirectory(const fs::path& dirPath, const string& newDirName)
{
    if (fs::exists(dirPath) && fs::is_directory(dirPath))
    {
        fs::path parentPath = dirPath.parent_path();
        fs::path newDirPath = parentPath / newDirName;

        try
        {
            fs::rename(dirPath, newDirPath);
            cout << "Directory renamed successfully." << endl;
        }
        catch (const fs::filesystem_error& error)
        {
            cout << "Failed to rename the directory: " << error.what() << endl;
        }
    }
    else
    {
        cout << "Invalid directory path." << endl;
    }

    _getch();
}

void copyFile(const fs::path& sourceFilePath, const fs::path& destinationFilePath)
{
    try
    {
        fs::copy(sourceFilePath, destinationFilePath);
        cout << "File copied successfully." << endl;
    }
    catch (const fs::filesystem_error& error)
    {
        cout << "Failed to copy the file: " << error.what() << endl;
    }

    _getch();
}

void copyDirectory(const fs::path& sourceDirPath, const fs::path& destinationDirPath)
{
    try
    {
        fs::copy(sourceDirPath, destinationDirPath, fs::copy_options::recursive);
        cout << "Directory copied successfully." << endl;
    }
    catch (const fs::filesystem_error& error)
    {
        cout << "Failed to copy the directory: " << error.what() << endl;
    }

    _getch();
}

void moveFile(const fs::path& sourceFilePath, const fs::path& destinationFilePath)

```

```
{
    try
    {
        fs::rename(sourceFilePath, destinationFilePath);
        cout << "File moved successfully." << endl;
    }
    catch (const fs::filesystem_error& error)
    {
        cout << "Failed to move the file: " << error.what() << endl;
    }

    _getch();
}

void moveDirectory(const fs::path& sourceDirPath, const fs::path& destinationDirPath)
{
    try
    {
        fs::rename(sourceDirPath, destinationDirPath);
        cout << "Directory moved successfully." << endl;
    }
    catch (const fs::filesystem_error& error)
    {
        cout << "Failed to move the directory: " << error.what() << endl;
    }

    _getch();
}

void absolutePath(fs::path& enteredPath, const fs::path& currentPath) {
    if (!enteredPath.has_root_path()) {
        enteredPath = currentPath / enteredPath;
    }
}

int main()
{
    fs::path currentPath = "D:\\"; // Вказати шлях до жорсткого диска (наприклад, "C:\\")

    vector<string> options = { "Exit", "List Contents", "Change Directory", "Create File",
    "Create Directory", "Delete File", "Delete Directory", "File Properties", "Open File", "Rename
File", "Rename Directory", "Copy File", "Copy Directory", "Move File", "Move Directory", "Here
are also supported commands cd and ls" };
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t Hello!\n\t\t\t\t\t\t\t Welcome to File
Explorer" << endl;
    _getch();

    while (true)
    {
        printMenu(options, currentPath);

        string inputst;
        getline(cin, inputst);
        clearScreen();
        if (inputst == "1") // Команда для закінчення роботи
        {
            break;
        }
        else if (inputst == "2" || inputst == "ls") // Вивід інформації про вміст за
поточним розташуванням користувача
        {
            displayDirectoryContents(currentPath);
            _getch();
        }
    }
}
```

```

}
else if (inputst == "3") // змінити дерикторію (вводити лише абсолютний шлях)
{
    string newPath;
    cout << "Enter absolute directory path: ";
    getline(cin, newPath);

    if (fs::exists(newPath) && fs::is_directory(newPath))
    {
        currentPath = newPath;
    }
    else
    {
        cout << "Invalid directory path." << endl;
        _getch();
    }
}
else if (inputst == "4") // Створення файлу
{
    string filePath;
    cout << "Enter file path: ";
    getline(cin, filePath);
    //отримання абсолютного шляху до файлу
    fs::path formattedPath = filePath;
    absolutePath(formattedPath, currentPath);
    createFile(formattedPath);
}
else if (inputst == "5") // Створення дерикторії
{
    string dirPath;
    cout << "Enter directory path: ";
    getline(cin, dirPath);
    fs::path formattedPath = dirPath;
    absolutePath(formattedPath, currentPath);

    createDirectory(formattedPath);
}
else if (inputst == "6") // видалення файлу
{
    string filePath;
    cout << "Enter file path: ";
    getline(cin, filePath);
    //отримання абсолютного шляху до файлу
    fs::path formattedPath = filePath;
    absolutePath(formattedPath, currentPath);

    deleteFile(formattedPath);
}
else if (inputst == "7") // видалення дерикторії
{
    string dirPath;
    cout << "Enter directory path: ";
    getline(cin, dirPath);
    fs::path formattedPath = dirPath;
    absolutePath(formattedPath, currentPath);

    deleteDirectory(formattedPath);
}
else if (inputst == "8") // отримати властивості файлу
{
    string path;
    cout << "Enter file path: ";
    getline(cin, path);
    fs::path formattedPath = path;
    absolutePath(formattedPath, currentPath);
}

```

```

        displayFileProperties(formattedPath);
    }
    else if (inputst == "9") // відкрити файл
    {
        string filePath;
        cout << "Enter file path: ";
        getline(cin, filePath);
        fs::path formattedPath = filePath;
        absolutePath(formattedPath, currentPath);

        openFile(formattedPath);
    }
    else if (inputst == "10") // перейменування файлу
    {
        string filePath;
        cout << "Enter file path: ";
        getline(cin, filePath);
        fs::path formattedPath = filePath;
        absolutePath(formattedPath, currentPath);

        string newFileName;
        cout << "Enter new file name: ";
        getline(cin, newFileName);

        renameFile(formattedPath, newFileName);
    }
    else if (inputst == "11") // перейменування дерикторії
    {
        string dirPath;
        cout << "Enter directory path: ";
        getline(cin, dirPath);
        fs::path formattedPath = dirPath;
        absolutePath(formattedPath, currentPath);

        string newDirName;
        cout << "Enter new directory name: ";
        getline(cin, newDirName);

        renameDirectory(formattedPath, newDirName);
    }
    else if (inputst == "12") // скопіювати файл
    {
        string sourceFilePath;
        cout << "Enter source file path: ";
        getline(cin, sourceFilePath);
        fs::path formattedPath = sourceFilePath;
        absolutePath(formattedPath, currentPath);

        string destinationFilePath;
        cout << "Enter absolute destination file path: ";
        getline(cin, destinationFilePath);

        copyFile(formattedPath, destinationFilePath);
    }
    else if (inputst == "13") // скопіювати дерикторію
    {
        string sourceDirPath;
        cout << "Enter source directory path: ";
        getline(cin, sourceDirPath);
        fs::path formattedPath = sourceDirPath;
        absolutePath(formattedPath, currentPath);

        string destinationDirPath;
        cout << "Enter absolute destination directory path: ";

```

```

        getline(cin, destinationDirPath);

        copyDirectory(formattedPath, destinationDirPath);
    }
    else if (inputst == "14") // Перемістити файл
    {
        string sourceFilePath;
        cout << "Enter source file path: ";
        getline(cin, sourceFilePath);
        fs::path formattedPath = sourceFilePath;
        absolutePath(formattedPath, currentPath);

        string destinationFilePath;
        cout << "Enter absolute destination file path and specify the file name at
the end of this path: ";
        getline(cin, destinationFilePath);

        moveFile(formattedPath, destinationFilePath);
    }
    else if (inputst == "15") // Перемістити дерикторію
    {
        string sourceDirPath;
        cout << "Enter source directory path: ";
        getline(cin, sourceDirPath);
        fs::path formattedPath = sourceDirPath;
        absolutePath(formattedPath, currentPath);

        string destinationDirPath;
        cout << "Enter absolute destination directory path and specify the
directory name at the end of this path: ";
        getline(cin, destinationDirPath);

        moveDirectory(formattedPath, destinationDirPath);
    }
    else if (inputst.find("cd") != string::npos && inputst.find("cd") == 0 &&
inputst.length() > 2) { //cd команда
        string extractedPath = inputst.substr(3);
        fs::path formattedPath;
        if (extractedPath == "..") {
            fs::path parentPath = currentPath.parent_path();
            formattedPath = parentPath;
        }
        else
            formattedPath = extractedPath;
        absolutePath(formattedPath, currentPath);
        if (fs::exists(formattedPath) && fs::is_directory(formattedPath)) {
            fs::path canonicalPath = fs::canonical(formattedPath);
            cout << "Now your current location path is " << canonicalPath <<
endl;

            currentPath = canonicalPath;
        }
        else
            cout << "Invalid directory path." << endl;

        getchar();
    }
}
return 0;
}

```