**PseudoCodes for this File Explorer Application:**

A list of pseudocodes for every day's work in the File Explorer Application project is provided below. Before you write the actual code, these pseudocodes help you comprehend the program's flow by outlining its general structure and logic.

**Design the Application Structure and Basic File Operations**

Objective:

Design the basic structure of the application.

Implement listing files in a directory.

1. Start

2. Define class FileExplorer

- Define attribute currentDirectory

- Define method listFiles()

- Get list of files and directories in currentDirectory

- Print each file and directory name

3. In main()

- Initialize FileExplorer object

- Set currentDirectory to a default path (e.g., "/home/user")

- Call listFiles() method to display files and directories

4. End

**Implement Directory Navigation**

Objective:

Allow the user to navigate through directories

1. Start

2. In FileExplorer class, add method changeDirectory(newPath)

- Check if newPath is a valid directory

- If valid, update currentDirectory to newPath

- If invalid, print error message

3. In main()

- Prompt user for input: 'list', 'cd <directory\_name>', 'exit'

- If 'list', call listFiles() method

- If 'cd <directory\_name>', call changeDirectory(newPath) with directory\_name

- If 'exit', terminate the program

4. Loop until 'exit' command is received

5. End

**File Manipulation Capabilities**

Objective:

Implement features to copy, move, delete, and create files.

1. Start

2. In FileExplorer class, add methods:

- copyFile(sourcePath, destinationPath)

- Copy file from sourcePath to destinationPath

- Print success or error message

- moveFile(sourcePath, destinationPath)

- Move file from sourcePath to destinationPath

- Print success or error message

- deleteFile(filePath)

- Delete file at filePath

- Print success or error message

- createFile(filePath)

- Create a new empty file at filePath

- Print success or error message

3. In main()

- Prompt user for input: 'copy <src> <dest>', 'move <src> <dest>', 'delete <file>', 'create <file>'

- Call the corresponding method based on user input

4. Loop until 'exit' command is received

5. End

**Implement File Search Functionality**

Objective:

Add the ability to search for files within the current directory.

1. Start

2. In FileExplorer class, add method searchFiles(keyword)

- Search for files and directories in currentDirectory matching keyword

- Print names of matching files and directories

3. In main()

- Prompt user for input: 'search <keyword>'

- Call searchFiles(keyword) with the user-provided keyword

4. Loop until 'exit' command is received

5. End

**Add File Permission Management Features**

Objective:

Allow the user to view and change file permissions.

1. Start

2. In FileExplorer class, add methods:

- viewPermissions(filePath)

- Get and display file permissions for file at filePath

- changePermissions(filePath, permissions)

- Change file permissions for file at filePath to the given permissions

- Print success or error message

3. In main()

- Prompt user for input: 'viewperm <file>', 'chmod <file> <permissions>'

- If 'viewperm <file>', call viewPermissions(filePath)

- If 'chmod <file> <permissions>', call changePermissions(filePath, permissions)

4. Loop until 'exit' command is received

5. End

**Overall Application Structure**

**Main Structure:**

1. Define class FileExplorer with necessary attributes and methods

2. Implement day-wise functionalities step by step

3. In main()

- Set up a loop to interact with the user

- Parse user commands and call appropriate methods in FileExplorer class

- Loop until the user enters 'exit'

This pseudocode acts as an implementation roadmap for the C++ console-based File Explorer Application. This can serve as a model for writing the C++ code for each task.