Console-Based File Explorer Application in C++

(Pseudo Code)

OVERVIEW:

Pseudo code

Explanation

* Function definition
* Pseudo code explanation
* END of the function

Start of Application

START

// Initialize necessary variables

SET isRunning = true

DECLARE command AS STRING

DECLARE currentPath AS STRING = "."

// Application main loop

WHILE (isRunning)

CALL displayMenu() // Show available commands to the user

SET command = getUserCommand() // Get command from the user

// Handle the command

IF command == "exit"

SET isRunning = false // End the loop to exit the application

ELSE

CALL processCommand(command, currentPath) // Process the command

ENDIF

ENDWHILE

// Display exit message

PRINT "Exiting application."

END

EXPLANATION:

1. **Initialize necessary variables**

* **SET isRunning = true**:
  + This variable (isRunning) is a boolean flag used to control the application's main loop.
  + It is set to true initially, meaning the application will keep running.
* **DECLARE command AS STRING**:
  + This variable (command) is declared as a string. It will hold the command that the user enters.
* **DECLARE currentPath AS STRING = "."**:
  + This variable (currentPath) is declared as a string and initialized to ".", which represents the current directory in a file system.
  + The program will use this as the starting point for file and directory operations.

**2. Application main loop**

* **WHILE (isRunning)**:
  + This is a loop that continues to execute as long as isRunning is true.
  + The loop ensures that the application keeps running until the user decides to exit.
* **CALL displayMenu()**:
  + This line calls the displayMenu() function, which is responsible for showing the available commands to the user.
  + This could be something like "ls" to list files, "cd" to change directories, etc.
* **SET command = getUserCommand()**:
  + This line calls the getUserCommand() function, which captures the user’s input and stores it in the command variable.
  + The user’s input determines what the application will do next.

**3. Handle the command**

* I**F command == "exit"**:
  + This condition checks if the user has entered the command "exit".
* **SET isRunning = false**:
  + If the command is "exit", the application sets isRunning to false.
  + This causes the WHILE loop to terminate, effectively ending the application.
* **ELSE**:
  + If the command is not "exit", the application proceeds to the next step.
* **CALL processCommand(command, currentPath)**:
  + This line calls the processCommand() function, passing the user’s command and the current directory path (currentPath) as arguments.
  + The processCommand() function will handle different commands like listing files, creating directories, etc., based on what the user entered.

**4. End of Loop and Exit**

**ENDWHILE**:

* This marks the end of the WHILE loop.
* Once the loop ends (when isRunning is false), the application will move on to the final steps.

**PRINT "Exiting application."**:

* This line prints a message to the user indicating that the application is exiting.
* It provides feedback to the user that the program is closing down.

DISPLAY MENU

**displayMenu()** is a function designed to show a menu of options to the user in a file explorer application.

FUNCTION displayMenu()

PRINT "File Explorer Menu:"

PRINT "1. List directory contents (ls)"

PRINT "2. Create a file (createfile)"

PRINT "3. Create a directory (createdir)"

PRINT "4. Rename a file/directory (rename)"

PRINT "5. Delete a file/directory (delete)"

PRINT "6. View file contents (viewfile)"

PRINT "7. Change directory (cd)"

PRINT "Type 'exit' to quit the application."

ENDFUNCTION

EXPLANATION:

Function Definition:

**FUNCTION displayMenu()**:

* This line defines a function called displayMenu.
* It does not take any parameters.
* Its purpose is to display a menu to the user with available commands.

Print Menu Header:

**PRINT "File Explorer Menu:"**:

* This line prints a header or title for the menu.
* It indicates that the following options are related to the file explorer application.

Print Menu Options:

PRINT "Type 'exit' to quit the application:

End of Function:

This line marks the end of the displayMenu() function.

It signifies that the function’s logic is complete and the function is now ready for use.

Get User Command

This pseudo code describes a function named getUserCommand() that is responsible for capturing and returning a command entered by the user

PSEUDO CODE:

FUNCTION getUserCommand() RETURNS STRING

DECLARE command AS STRING

PRINT "Enter command: "

INPUT command

RETURN command

ENDFUNCTION

Explanation

**FUNCTION getUserCommand()**:

* This line defines a function named getUserCommand.
* The purpose of this function is to encapsulate the logic for obtaining a command from the user.

**RETURNS STRING**:

* This specifies the return type of the function, which is STRING.
* The function will return a string that contains the command the user entered.

Process Command

The pseudo code provided defines a function named processCommand() that processes user commands to perform various file operations.

PSEUDO CODE

FUNCTION processCommand(command, currentPath)

IF command == "ls"

CALL listDirectory(currentPath)

ELSE IF command == "createfile"

DECLARE path AS STRING

PRINT "Enter file path: "

INPUT path

CALL createFile(path)

ELSE IF command == "createdir"

DECLARE path AS STRING

PRINT "Enter directory path: "

INPUT path

CALL createDirectory(path)

ELSE IF command == "rename"

DECLARE oldPath AS STRING

DECLARE newPath AS STRING

PRINT "Enter old path: "

INPUT oldPath

PRINT "Enter new path: "

INPUT newPath

CALL renameItem(oldPath, newPath)

ELSE IF command == "delete"

DECLARE path AS STRING

PRINT "Enter path: "

INPUT path

CALL deleteItem(path)

ELSE IF command == "viewfile"

DECLARE path AS STRING

PRINT "Enter file path: "

INPUT path

CALL viewFile(path)

ELSE IF command == "cd"

DECLARE path AS STRING

PRINT "Enter directory path: "

INPUT path

CALL changeDirectory(path, currentPath)

ELSE

PRINT "Invalid command!"

ENDIF

ENDFUNCTION

Explanation

**FUNCTION processCommand(command, currentPath)**:

* This defines a function called processCommand.
* It takes two parameters:
  + command: A string that represents the user’s command.
  + currentPath: A string representing the current directory path where operations are performed

1.Handle "ls" Command

**IF command == "ls"**:

* Checks if the command entered by the user is "ls", which typically stands for "list directory contents".

**CALL listDirectory(currentPath)**:

* Calls the listDirectory() function, passing currentPath as an argument.
* This function lists all files and directories in the current directory.

2. Handle "createfile" Command

**ELSE IF command == "createfile"**:

* Checks if the command is "createfile", which is used to create a new file.

**DECLARE path AS STRING**:

* Declares a variable path to store the path where the file will be created.

**PRINT "Enter file path: "**:

* Prompts the user to enter the file path.

**INPUT path**:

* Captures the user’s input and stores it in the path variable.

**CALL createFile(path)**:

* Calls the createFile() function with the path variable as an argument.
* This function creates a new file at the specified path.

3.Handle "rename" Command

**DECLARE oldPath AS STRING** and **DECLARE newPath AS STRING**:

* Declares two variables: oldPath for the current path of the item to be renamed and newPath for the new path.

**PRINT "Enter old path: "** and **PRINT "Enter new path: "**:

* Prompts the user to enter the old and new paths.

**INPUT oldPath** and **INPUT newPath**:

* Captures the user’s input for the old and new paths.

**CALL renameItem(oldPath, newPath)**:

* Calls the renameItem() function with oldPath and newPath as arguments.
* This function renames the item from oldPath to newPath.

**ELSE IF command == "createdir"**:

* Checks if the command is "createdir", which is used to create a new directory.

**DECLARE path AS STRING**:

* Declares a variable path to store the path where the directory will be created.

**PRINT "Enter directory path: "**:

* Prompts the user to enter the directory path.

**INPUT path**:

* Captures the user’s input and stores it in the path variable.

**CALL createDirectory(path)**:

* Calls the createDirectory() function with the path variable as an argument.
* This function creates a new directory at the specified path.

**5. Handle "delete" Command**

**ELSE IF command == "delete"**:

* Checks if the command is "delete", which is used to delete a file or directory.

**DECLARE path AS STRING**:

* Declares a variable path to store the path of the item to be deleted.

**PRINT "Enter path: "**:

* Prompts the user to enter the path of the item to be deleted.

**INPUT path**:

* Captures the user’s input and stores it in the path variable.

**CALL deleteItem(path)**:

* Calls the deleteItem() function with the path variable as an argument.
* This function deletes the file or directory at the specified path.

6.Handle "viewfile" Command

**ELSE IF command == "viewfile"**:

* Checks if the command is "viewfile", which is used to view the contents of a file.

**DECLARE path AS STRING**:

* Declares a variable path to store the path of the file to be viewed.

**PRINT "Enter file path: "**:

* Prompts the user to enter the file path.

**INPUT path**:

* Captures the user’s input and stores it in the path variable.

**CALL viewFile(path)**:

* Calls the viewFile() function with the path variable as an argument.
* This function displays the contents of the file at the specified path

7.Handle "cd" Command

**ELSE IF command == "cd"**:

* Checks if the command is "cd", which is used to change the current directory.

**DECLARE path AS STRING**:

* Declares a variable path to store the path of the directory to which the user wants to change.

**PRINT "Enter directory path: "**:

* Prompts the user to enter the directory path.

**INPUT path**:

* Captures the user’s input and stores it in the path variable.

**CALL changeDirectory(path, currentPath)**:

* Calls the changeDirectory() function with the new path and the current path as arguments.
* This function changes the current working directory to the specified path.

8.Handle Invalid Command

**ELSE**:

* Catches any command that doesn't match the predefined options.

**PRINT "Invalid command!"**:

* Outputs a message indicating that the entered command is not recognized.
* This helps the user understand that they need to enter a valid command.

**LIST DIRECTORY CONTENTS**

PSEUDO CODE:

FUNCTION listDirectory(path)

IF directory exists at path AND is a directory

FOR EACH item IN directory at path

PRINT item.name

ENDFOR

ELSE

PRINT "Directory does not exist"

ENDIF

ENDFUNCTION

EXPLANATION:

1. Check Directory Existence( IF directory exists at path AND is a directory)

* This condition checks if the directory at the specified path exists and if it is indeed a directory.
* It ensures that the function only proceeds if the provided path is valid and points to a directory.
* If this condition is true, the function will continue to list the directory contents.
* If not, the function will handle the case where the directory does not exist or is not a directory.

2.List Directory Contents

**FOR EACH item IN directory at path**:

* This loop iterates over each item (file or subdirectory) in the directory specified by path.
* The loop variable item represents each individual item within the directory.

**PRINT item.name**:

* Inside the loop, this line prints the name of each item.
* item.name refers to the name of the current item being processed in the loop.

**ENDFOR**:

* This marks the end of the FOR EACH loop.
* After listing all items, the loop terminates.

3.Handle Directory Not Existing

* **ELSE**:
  + This part of the code handles the situation where the directory does not exist or the path is not valid.
* **PRINT "Directory does not exist"**:
  + This line outputs a message to the user indicating that the directory specified by path could not be found or is not a directory.
* **ENDIF**:
  + This marks the end of the IF condition block.
  + It concludes the conditional logic used to check the directory's existence.

**End of Function**

This signifies the end of the listDirectory() function.

It marks the completion of the function’s logic.

**CREATE A NEW FILE**

The pseudo code defines a function named createFile() that creates a new file at a specified path.

PSEUDO CODE:

FUNCTION createFile(path)

IF file does not exist at path

CREATE an empty file at path

PRINT "File created successfully"

ELSE

PRINT "File already exists"

ENDIF

ENDFUNCTION

EXPLANATION:

**FUNCTION createFile(path)**:

* This line defines a function named createFile.
* It takes one parameter, path, which is a string representing the path where the new file will be created.

1.Check if the File Exists

**IF file does not exist at path**:

* This condition checks if there is no file currently present at the specified path.
* It ensures that the function will only proceed with creating the file if a file does not already exist at that location.
* If this condition is true, meaning the file does not exist, the function will move to the next step to create it.
* If the file already exists, the function will handle this scenario separately.

1. Create a New File

**CREATE an empty file at path**:

* This action creates a new, empty file at the specified path.
* The file is created without any initial content.
* The exact method for creating a file depends on the implementation but generally involves opening a file for writing, which creates it if it doesn't already exist.

**PRINT "File created successfully"**:

* This line provides feedback to the user, indicating that the file has been created successfully.
* It lets the user know that their request to create a file has been fulfilled.

1. Handle File Already Exists

**ELSE**:

* This part of the code is executed if the condition that checks for the file's existence is false.
* It means that a file already exists at the specified path.

**PRINT "File already exists"**:

* This line outputs a message to the user indicating that a file already exists at the given path.
* It informs the user that the file was not created because it already exists.

End of Function:

* + This line marks the end of the createFile() function.
  + It signifies that the function’s logic is complete and that the function is now ready for use.

**CREATE A NEW DIRECTORY**

**The pseudo code defines a function named createDirectory() that creates a new directory at a specified path.**

**PSEUDO CODE**

FUNCTION createDirectory(path)

IF directory does not exist at path

CREATE a new directory at path

PRINT "Directory created successfully"

ELSE

PRINT "Directory already exists"

ENDIF

ENDFUNCTION

EXPLANATION:

Function Definition:

**FUNCTION createDirectory(path)**:

* This line defines a function named createDirectory.
* It takes one parameter, path, which is a string representing the location where the new directory will be created.

1.Check if the Directory Exists:

**IF directory does not exist at path**:

* This condition checks whether there is no directory currently at the specified path.
* It ensures that the function only proceeds with creating the directory if a directory does not already exist at that location.
* If this condition is true (i.e., no directory exists at the path), the function will move on to create the directory.
* If the directory already exists, the function will handle this scenario separately.

2. Create a New Directory

**CREATE a new directory at path**:

* This action creates a new directory at the specified path.
* The directory is created with no initial contents.
* The exact method for creating a directory depends on the implementation but generally involves system calls or functions designed to create directories.

**PRINT "Directory created successfully"**:

* This line outputs a message to the user, indicating that the directory has been created successfully.
* It provides feedback to the user that their request to create a directory has been fulfilled.

3.Handle Directory Already Exists

**ELSE**:

* This part of the code is executed if the condition that checks for the directory’s existence is false.
* It means that a directory already exists at the specified path.

**PRINT "Directory already exists"**:

* This line outputs a message to the user indicating that a directory already exists at the given path.
* It informs the user that the directory was not created because it already exists.

End of Function:

This line marks the end of the createDirectory() function.

It signifies that the function’s logic is complete and that the function is now ready for use.

**DELETE A FILE OR DIRECTORY**

The pseudo code defines a function named deleteItem() that deletes a file or directory at a specified path.

PSEUDO CODE

FUNCTION deleteItem(path)

IF item exists at path

DELETE item at path

PRINT "Item deleted successfully"

ELSE

PRINT "Item does not exist"

ENDIF

ENDFUNCTION

EXPLANATION:

**FUNCTION deleteItem(path)**:

* This line defines a function called deleteItem.
* It takes one parameter, path, which is a string representing the path to the file or directory that is to be deleted.

Check if the Item Exists:

* **IF item exists at path**:
  + This condition checks if there is an item (either a file or a directory) at the specified path.
  + It ensures that the function only proceeds with deleting the item if it actually exists.
  + If the condition is true (i.e., the item exists at the path), the function will continue with the deletion process.
  + If the item does not exist, the function will handle this scenario separately.

Delete the Item:

**DELETE item at path**:

* This action deletes the item (file or directory) at the specified path.
* The exact method for deleting an item depends on the implementation, but generally involves system calls or functions designed to remove files or directories.

**PRINT "Item deleted successfully"**:

* This line provides feedback to the user, indicating that the item has been successfully deleted.
* It informs the user that their request to delete the item has been processed.

Handle Item Does Not Exist

**ELSE**:

* This part of the code is executed if the condition checking for the item's existence is false.
* It means that no item exists at the specified path.

**PRINT "Item does not exist"**:

* This line outputs a message to the user indicating that there is no item to delete at the given path.
* It informs the user that the deletion could not be performed because the item does not exist.

**TO VIEW FILE CONTENT**

**The pseudo code defines a function named viewFile() that reads and displays the contents of a file at a specified path.**

PSEUDO CODE

FUNCTION viewFile(path)

IF file exists at path AND is a regular file

OPEN file at path for reading

FOR EACH line in file

PRINT line

ENDFOR

CLOSE file

ELSE

PRINT "File does not exist"

ENDIF

ENDFUNCTION

EXPLANATION:

Function Definition:

**FUNCTION viewFile(path)**:

* This line defines a function called viewFile.
* It takes one parameter, path, which is a string representing the path to the file that will be viewed.

Check if the File Exists and is a Regular File:

**IF file exists at path AND is a regular file**:

* This condition checks whether a file exists at the specified path and ensures that it is a regular file (not a directory or special file).
* It verifies that the file can be opened and read.
* If the condition is true (i.e., the file exists and is a regular file), the function will proceed to open and read the file.
* If the file does not exist or is not a regular file, the function will handle this situation separately.

Open the File and Read Its Contents:

**OPEN file at path for reading**:

* This action opens the file located at the specified path in read mode.
* The file is prepared for reading its contents.

**FOR EACH line in file**:

* This loop iterates over each line in the opened file.
* The loop variable line represents the current line being read from the file.

**PRINT line**:

* Inside the loop, this line prints the content of each line to the console.
* It displays the contents of the file line by line.

**ENDFOR**:

* This marks the end of the FOR EACH loop.
* After reading and printing all lines, the loop terminates.

**CLOSE file**:

* This action closes the file after reading is complete.
* It releases any resources associated with the file and ensures that no more read operations can be performed

Handle File Does Not Exist:

**ELSE**:

* This part of the code is executed if the condition checking for the file’s existence and regularity is false.
* It means that the file either does not exist or is not a regular file.

**PRINT "File does not exist"**:

* This line outputs a message to the user indicating that the file could not be found or is not a regular file.
* It informs the user that the requested file cannot be viewed because it does not meet the expected criteria.

End of Function:

* This line marks the end of the viewFile() function.
* It signifies that the function’s logic is complete and that the function is now ready for use.

**Change Current Directory**

**The pseudo code defines a function named changeDirectory() that changes the current working directory to a specified path.**

PSEUDO CODE

FUNCTION changeDirectory(path, currentPath)

IF directory exists at path AND is a directory

SET currentPath = path

PRINT "Directory changed successfully"

ELSE

PRINT "Directory does not exist"

ENDIF

ENDFUNCTION

EXPLANATION:

Function definition:

**FUNCTION changeDirectory(path, currentPath)**:

* This line defines a function called changeDirectory.
* It takes two parameters:
  + path: A string representing the path to which the directory should be changed.
  + currentPath: A variable that holds the current working directory.

Check if the Directory Exists and is a Directory:

**IF directory exists at path AND is a directory**:

* + This condition checks if a directory exists at the specified path and ensures that it is indeed a directory (not a file or other type of item).
  + It verifies that the provided path is valid and points to a directory.
  + If this condition is true (i.e., the directory exists and is valid), the function will proceed to change the current directory.
  + If the directory does not exist or is not valid, the function will handle this situatation separately.

Change the Current Directory:

* **SET currentPath = path**:
  + This action updates the currentPath variable to the new directory path specified by path.
  + It effectively changes the working directory to the new location.
* **PRINT "Directory changed successfully"**:
  + This line provides feedback to the user, indicating that the directory has been successfully changed.
  + It informs the user that the current directory has been updated to the new path.

**Handle Directory Does Not Exist**

* **ELSE**:
  + This part of the code is executed if the condition checking for the directory's existence and validity is false.
  + It means that no valid directory exists at the specified path.
* **PRINT "Directory does not exist"**:
  + This line outputs a message to the user indicating that the specified directory could not be found or is not valid.
  + It informs the user that the attempt to change the directory was unsuccessful due to the directory not existing.

**End of Function**

This line marks the end of the changeDirectory() function.

It signifies that the function’s logic is complete and that the function is now ready for use.