# Code Sample: FileSystem

## *INTRODUCTION*:

This project is based to analyze the directory structure on the local machine,

- looking for duplicate file names
- Searching for text in specified files.

The C++ standard libraries provide a rich set of containers to support structuring data for this analysis. However, there is a surprising omission in the standard libraries – there is no support for managing directory information.

- This is an Academic Project for the Object Oriented Design Coursework. You will see in this project software modelling is implemented.
- The project 'FileSystem' is divided into different Packages (modules) which clearly demarcate into specific task performed by each package and to reduce tight coupling between the classes
- All the Object Oriented Principles have been showcased.
- It has been implemented in C++ 11.

## *DESCRIPTION:*

The following are the packages that have been implemented in the project:

- **Datastore**: a catalog of all files in the file set, saving each file name only once and saving each path only once, while preserving all of the containment relationships between directories and their files.

- **Executive**-it acts like a central module which delegates the work and calls the required classes for doing various tasks.
  Based on the input given by user, the information is parsed and information is given to the fileMgr class to get the required files. Furtherly based on requirement, the information is displayed using Display class.

- **FileMgr**: This package based on the input given, the path is parsed to get the set of files to be analyzed. It acts as a File Manager.

- **Display**: This package helps in displaying the information according to the user inputs parsed

## *OUTPUT GUIDELINES:*

The following where the output guidelines that were needed to be reproduced:

1. 1)Identify a set of files for analysis by supplying, on the command line, a path, one or more file patterns, and a switch **/s** which, if present, indicates that the entire directory tree rooted at the path is searched for matching files. If the switch is not present on the command line only the directory at that path is searched.

   Typical Format: Debug\Executive.exe TestFolder *.cpp /s

2. Support the use of a command line option /d that, when present, causes your program to emit a list of duplicate file names along with their paths.

   Typical Format: Debug\Executive.exe TestFolder *.h /s /d

3. Provide a command line option, **/f**<search text> which, when present, causes your program to list all the files stored in the catalog that contain the search text3.

   Typical Format: Debug\Executive.exe TestFolder *.h /d /fKusum

4. If no options are provided on the command line, emit a brief summary, e.g., N Files found in M directories.

   Typical Format: Debug\Executive.exe TestFolder

5. After construction of the catalog and emitting any specified results, accept from the console new text specifications for text searches in the catalog by providing text and file pattern(s). No other commands are to be accepted.

Typical Format:Enter any text to search:"Kusum" *.h
(Currently only implemented for .h extension files)

**ALL THE SAMPLE TEST CASES HAVE BEEN PROVIDED IN 'RUN.BAT' FILE.**

## *CONFIGURATION DETAILS:*

- I have used Visual Studio to code the project.
- There is a 'run.bat' file which can be used to run to see the result.
- All the test cases have been provided in the run.bat file. One can modify the run.bat file to test various other options.
- Folder named 'Test Folder' which is used for testing purpose of this project.(This is a dummy File Directory structure to run the project on . It has various duplicate files and directory structure which has been used to demonstrate the functionality of the project).
- One can test the functionality of the code by running the 'run.bat' file in admin mode.

## *SCREENSHOT OF THE OUTPUT:*