CODE CRACKER

CAPSTONE PROJECT

MENTOR : SOURISH DAS GUPTA

CODE CRACKERS

OUR TEAM

Dipkumar Zadafiya 202301159

Tirth Kheni 202301007

Harsh Kakadiya 202301052

Jenish Vasani 202301057

PROBLEM

Folder Cleaner:

- Build a Folder Cleaner. It deletes files that are:
  1. Redundant files (i.e., a newer copy is present),
  2. Files old for more than N number of months,
  3. Empty files (i.e., no content),
  4. Files that have not been accessed for at least M number of times.

Comments:

- Input data will be a file (can be txt or xls) with records in the format:
  1. Folder name
  2. Files in the folder
  3. File created date

ALGORITHM OF CODE

1. Include necessary header files for input/output operations, file handling, string manipulation, and time functions.

2. Define a constant `MAX_FILES` to specify the maximum number of files to be processed.

3. Define a struct `FileData` to store information about each file, including its name, date, size, and access count.

4. Implement a function `isFileOlderThanNMonths` to check if a file is older than a specified number of months.

5. Implement a function deletefiles to delete files based on certain criteria such as age, size, redundancy, and access count. This function takes an array of FileData, the count of files, and the values of N (number of months) and M (minimum access count) as input parameters.

6. Inside the main function:
   - Use a do-while loop to repeatedly prompt the user for folder name, number of months (N), and minimum access count (M).
   - Open the specified folder and count the number of files in it.
   - Initialize an array of FileData structs to store information about each file.
   - Read file information (name, date, size, access count) from the folder and populate the array.
   - Close the folder.
   - Call the deletefiles function to delete files based on the specified criteria.
   - Open the folder again in write mode to overwrite its contents with the updated file information.
   - Write the updated file information to the folder.
   - Close the folder.
   - Print deletion reasons if any files were deleted.
   - Prompt the user to continue or exit the program based on their choice.

7. The program continues executing until the user chooses to exit by entering "no" when prompted.

DATA STRUCTURE CHOSEN

1. Struct FileData:
   - This structure is designed to store information about each file.
   - It includes fields such as file name, date of creation, size, and access count.
   - By encapsulating this information within a struct, it becomes easier to manage and manipulate file data.

2. Vector:
   - Vectors are used to dynamically store instances of `FileData`.
   - They provide flexibility in handling a variable number of files, allowing the program to adapt to different folder sizes.
   - The use of vectors enables efficient memory allocation and management, ensuring optimal performance during file processing.

3. String:

- Strings are employed to store file names and dates extracted from the input file.
- They facilitate the manipulation of textual data, allowing for parsing and comparison operations.
- String operations are essential for tasks such as identifying redundant files and constructing deletion reasons.

➤ By leveraging these data structures, the algorithm achieves a balance between memory efficiency and computational effectiveness, enabling robust management of files within the specified folder.

TIME COMPLEXITY

The Program reads the input file line by line to count lines, so the time complexity of this step is O(n), where n is the number of lines.

- Reading the file of array: After the number of lines the program reads, the program reads data from the file into an array of FileData structs. Thus it also has a time linear time complexity relative to the number of lines in the file,which is O(n).

- Processing Files:- processFile() another loop is used to check for newer versions of files by comparing each file with every other file. This result in nested loop with quadratic equation time complexity, denoted as O(n^2). The function sFileOlderThanNMonths() is called to determine if the file is older than the N months. Which is typically constant or logarithmic time, So overall time complexity for processing files becomes O(n^2).

SPACE COMPLEXITY

- Array to store file Data:- The program uses an array of FileData struct to store file data read from the input file, so its time complexity is O(n).

- String Storage:- String variables are used to store file names,temporary values and data during file processing. However, since they are temporary and reused, their total space complexity is not significant compared to the array storing file data.

- Remaining Variables:- Other variables such as N,M,count, and temporary variables used in processing have constant space complexity relative to the input size.

Therefore, overall time complexity and space complexity lead to the O(n^2) and O(n) respectively, where n is the number of lines.

THANK YOU