Assignment 2: File Crawler

Introduction and purpose

The goal is to familiarize yourself with how we can handle files and folders using recursion. The first bonus task teaches you to use a hash table in practice.

Task

The task is to write a simple local search program that takes in a text string from the user and performs a search in the directory where the program is running, as well as its subdirectories.
The program should loop through the contents of all files to see if the specified text string exists in the file.

- The program should start with a message telling the user to enter one keywords.
- If the program finds a file that contains the text string, the file's absolute path must be printed on a separate line.
- If the file does not contain the text string, then nothing should be printed for that file. • If the file (or a directory) cannot be read, an error message containing at least the absolute path of the file/directory is printed.

Tip
Create a new folder/project for the submission and put the files from "Assignment 3" in the teacher's repo.

File searching may be easier to achieve in Python than C++. In Python, you can create the program yourself from scratch (you want to use the "os" library). In C++, I will set up a basic structure with CMake to use Boost's filesystem library.
There are other ways to read files and file structure in C++ that you can use:

- fstream (C++ STL) (ifstream, ofstream) • C stdio.h: fopen, fread, fwrite, fclose

Bonus tasks General:
Implementation in Python and C++, up to 2 bonus points for all submission tasks. If you solve this task in both Python and C++, it is sufficient that at least one uses recursion.

Bonus task 1: (1.5 bonus points)

Make an alternative implementation of the task where you have a function that indexes the entire directory.
This indexing should then be able to be used to find words quickly:

- The indexing function must go through the entire directory structure and all files • For each file, it must index each word (split the file's text into words) and save in which file the word can be found in the hash table (the word as key and file paths as value)
- Keep in mind that the same word can be found in several files. The values in the hash table must therefore fit multiple file paths
- Hash table is an abstract data type. The actual data type you should use depends on the programming language.
- The main function must first call the indexing function. Then the usual search function. The search function should use the hash table to find and print the path to all files where the word occurs.

Note: You must make an implementation/function that follows the basic task as well. These can be in the same or different programming languages.

Bonus task 2 (Up to 1.5 bonus points)

Include arguments to your program, which you choose from the list below. Each implemented argument gives 0.5 bonus points (up to 3 arguments). (If you have any ideas of your own about arguments you want to implement, that might be ok, ask me). The search string itself can also be an option if you want or handled as user input. Bonus task 2 can be an extension of bonus task 1 or build on the basic task.

Example:

- Search Mode: Implement options to perform a case-sensitive or case-insensitive search.

The arguments to the program can then look like this:

```
$ file_crawler.exe -c # Performs case sensitive search $
file_crawler.exe -i # Performs case insensitive search
```

Or in python:

```
$ python file_crawler.py -c # Performs case sensitive search $ python
file_crawler.py -i # Performs case insensitive search
```

Other suggestions for arguments / options:

- File Types: Allow the user to specify the type of files to be searched (eg, .txt, .pdf, .docx, MD).
  $ file_crawler.exe --file-types=txt,md
- Output File: Provide an option to write the results to a file, instead of printing to the console. • Display line numbers: Also print line numbers • Search Depth: Set a limit on the number of subdirectories to be searched, so that the search doesn't go too deep. • Search in Sub-directories only • Options for recursive/ non-recursive: Set if the search shall be recursive or not (non recursive: Only search top folder)

- Delimiter: (If using index/hashtable). Allow user to specify delimiter for words to use • Exclude Directories: Allow the user to specify directories to be excluded from the search.

Accounting The

source code must be pushed to a separate repository on GitHub.

The submission takes place by sending a link to the gitrepot into Omniway.