Learning C
The following code was able to print my name in the console
Was also able to open and read my file in d:\Documents\Vs Code Projects\Coding Assignments\Group 1\school.c
Was also able to prints out the content in that .c file in the console

```c
#include<stdio.h>

int main()
{
    printf("hello Katchiets");

    FILE *file;
    char line[100]; // Assuming lines in the file are no longer than 100 characters.

    // Open the file for reading.
    file = fopen("D:\\Documents\\Vs Code Projects\\Coding Assignments\\Group 1\\school.c", "r");

    if (file == NULL)
    {
        printf("Error opening the file.\n");
        return 1;
    }

    // Read and print each line until the end of the file.
    while (fgets(line, sizeof(line), file) != NULL)
    {
        printf("%s", line);
    }

    // Close the file.
    fclose(file);
}
```

In C strings, backslashes () are used as escape characters to represent special characters. For example, the escape sequence **\n** is used to represent a newline character, **\t** for a tab character, etc. Because backslashes have this special meaning, if you want to include an actual backslash character in a C string, you need to escape it with another backslash.

Here's a brief explanation of how escaping works in C strings:

1. **Single Backslash**: If you want to include a single backslash in a C string, you need to escape it by using another backslash. For example, to represent a string containing a single backslash, you write **"\\\\"**. The first backslash escapes the second one, resulting in a single backslash in the actual string.

2. **Forward Slash**: Forward slashes (/) do not have a special meaning in C strings, so they don't need to be escaped. You can use forward slashes as regular characters in C strings. For example, **"folder/file.txt"** is a valid C string.

3. **Path Separators**: In Windows file paths, backslashes are used as path separators (e.g., **C:\Documents\file.txt**). However, since backslashes are escape characters in C strings, you need to use double backslashes (\) to represent a single backslash in the path. So, in C strings, a Windows file path would look like **"C:\\Documents\\file.txt"**.

Here's an example to illustrate this with a C string containing a Windows file path:

```c
#include <stdio.h>

int main()
{
    // A C string representing a Windows file path
    char path[] = "D:\\Documents\\Vs Code Projects\\Coding Assignments\\Group 1\\school.c";

    // Print the path to the console
    printf("File Path: %s\n", path);

    return 0;
}
```

In this example, the **path** variable contains a C string representing a Windows file path. We use double backslashes to escape each backslash, ensuring that the path is correctly represented as a C string. When we print the **path** variable using **printf()**, it will display the correct file path:

```
File Path: D:\Documents\Vs Code Projects\Coding Assignments\Group 1\school.c
```

Remember that if you're using forward slashes in your file path, you don't need to escape them. They can be used as regular characters in C strings. However, using double backslashes for path separators is a common practice to ensure portability and consistency across different platforms.

This failed execution

```c
#include <stdio.h>
#include <dirent.h>
#include <string.h>

void listDirectoriesAndFiles(const char *path);

int main()
{
    printf("Listing directories and files in D:\\Documents:\n");
    listDirectoriesAndFiles("D:\\Documents");
    return 0;
}

void listDirectoriesAndFiles(const char *path)
{
    DIR *dir;
    struct dirent *entry;
    char folders[1000][FILENAME_MAX]; // Array to store unique folder names
    int numFolders = 0; // Counter for the number of unique folders

    dir = opendir(path);
    if (dir == NULL)
    {
        printf("Error opening directory: %s\n", path);
        return;
    }

    // Loop through the directory entries and find unique folder names.
    while ((entry = readdir(dir)) != NULL)
    {
        // Skip the "." and ".." entries and non-directory entries.
        if (entry->d_type != DT_DIR || strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
            continue;

        int found = 0;
        // Check if the folder name is already in the list of unique folders.
        for (int i = 0; i < numFolders; i++)
        {
            if (strcmp(folders[i], entry->d_name) == 0)
            {
                found = 1;
                break;
            }
        }

        if (!found)
        {
            // If the folder name is not found in the list, add it to the list.
            strcpy(folders[numFolders], entry->d_name);
            numFolders++;
        }
    }

    closedir(dir);

    // Print the summary of unique folder names.
    printf("Folders present in D:\\Documents include (");
    for (int i = 0; i < numFolders; i++)
    {
        printf("%s", folders[i]);
        if (i < numFolders - 1)
            printf(", ");
    }
    printf(")\n");
}
```

The following code failed execution:

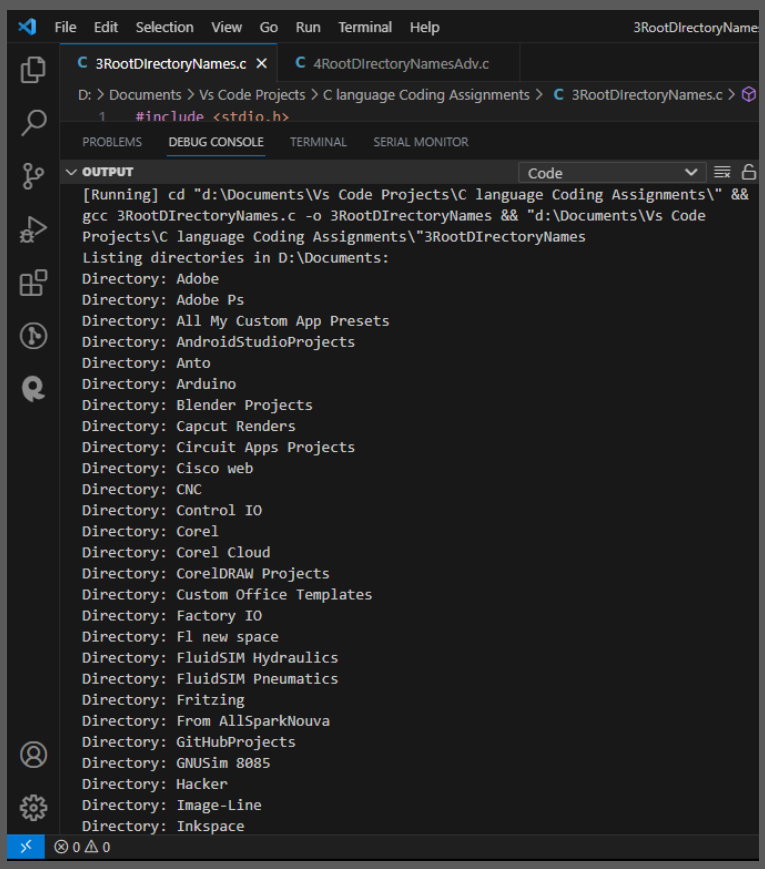
```
#include <stdio.h>
#include <windows.h>

void listDirectoriesInRoot(const wchar_t *path);

int main()
{
    printf("Listing directories in D:\\Documents:\n");
    listDirectoriesInRoot(L"D:\\Documents"); // Note the L before the string to create a wide-character string
    return 0;
}

void listDirectoriesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData; // Use the wide-character version of the structure
    HANDLE hFind = FindFirstFileW(path, &findFileData); // Use the wide-character version of the function
    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories in: %S\n", path); // Use %S to print wide-character strings
        return;
    }

    // Loop through the directory entries and find directories.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0); // Use the wide-character version of the function

    FindClose(hFind);
}
```

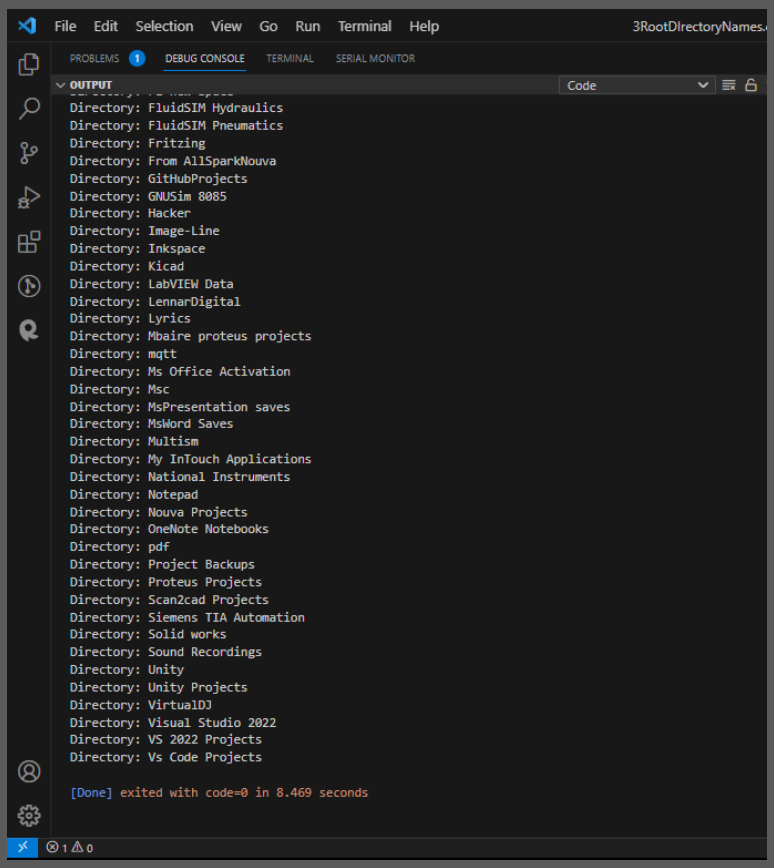
This also failed Execution

```
#include <stdio.h>
#include <windows.h>

void listDirectoriesInRoot(const char *path);

int main()
{
    printf("Listing directories in D:\\Documents:\n");
    listDirectoriesInRoot("D:\\Documents");
    return 0;
}

void listDirectoriesInRoot(const char *path)
{
    WIN32_FIND_DATA findFileData;
    HANDLE hFind = FindFirstFile((LPCWSTR)path, &findFileData);
    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories in: %s\n", path);
        return;
    }

    // Loop through the directory entries and find directories.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\n", findFileData.cFileName);
            }
        }
    } while (FindNextFile(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```

The following code was able to access my D:/Documents path and return all the directories in the root

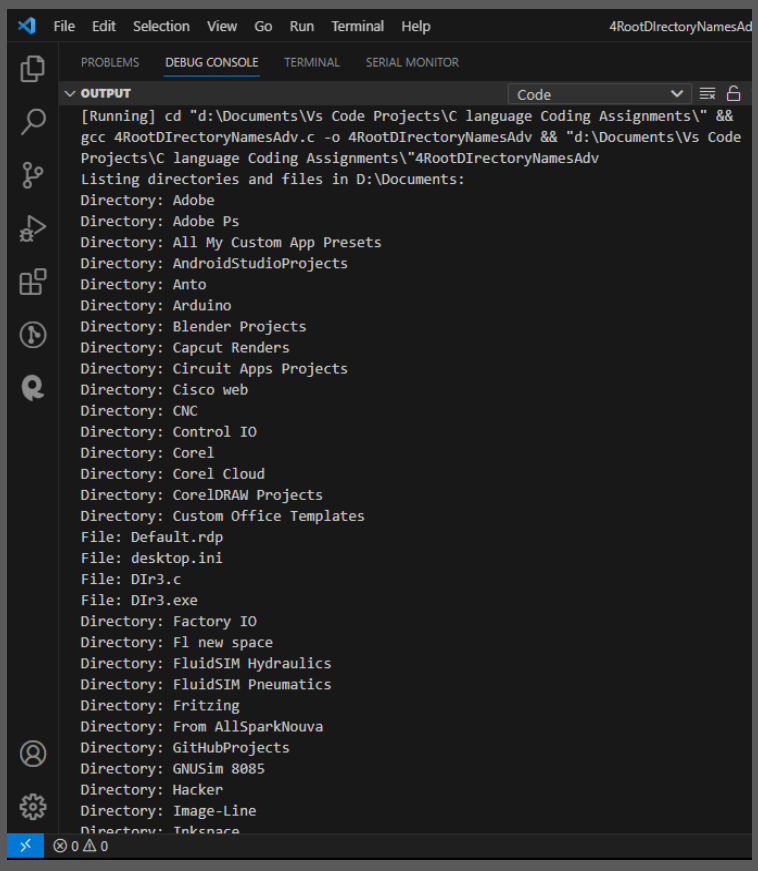
```c
#include <stdio.h>
#include <windows.h>

void listDirectoriesInRoot(const wchar_t *path);

int main()
{
    printf("Listing directories in D:\\Documents:\n");
    listDirectoriesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories in: %S\n", path);
        return;
    }

    // Loop through the directory entries and find directories.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```

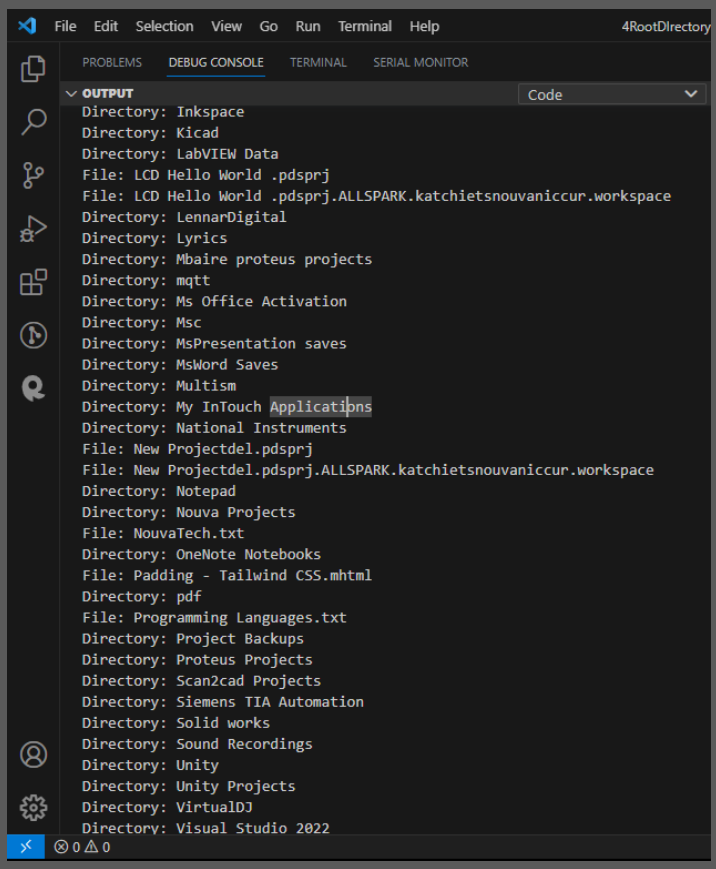




The following code returned me the files and directories present in the root but in alphabetical order:

```c
#include <stdio.h>
#include <windows.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    printf("Listing directories and files in D:\\Documents:\n");
    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\n", path);
        return;
    }

    // Loop through the directory entries and find directories and non-directory files.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\n", findFileData.cFileName);
            }
        }
        else
        {
            wprintf(L"File: %s\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```





gg

The following returned in a list form:

```
#include <stdio.h>
#include <windows.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    printf("Listing directories and files in D:\\Documents:\n");
    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            wprintf(L"File: %s\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```

```
OUTPUT                                                    Code
Listing directories and files in D:\Documents:
Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUSim 8085
Directory: Hacker
Directory: Image-Line
⊗ 0 ⚠ 0
Directory: Visual Studio 2022
Directory: VS 2022 Projects
Directory: Vs Code Projects
Files:
File: Default.rdp
File: desktop.ini
File: DIr3.c
File: DIr3.exe
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

[Done] exited with code=0 in 8.298 seconds
⊗ 0 ⚠ 0
```

Retrieving code path:

```
File  Edit  Selection  View  Go  Run  Terminal  Help        5RootDIrectoryNamesAdv.c - Visual Studio Code
C 3RootDIrectoryNames.c   C 4RootDIrectoryNamesAdv.c   C 5RootDIrectoryNamesAdv.c ×
D: > Documents > Vs Code Projects > C language Coding Assignments > C 5RootDIrectoryNamesAdv.c > main()
1   #include <stdio.h>
2   #include <windows.h>
3
4   void listDirectoriesAndFilesInRoot(const wchar_t *path);
5
6   int main()
7   {
8       // Print the current path location of the file being executed
9       printf("Listing directories and files in D:\\Documents (Current File Location: %s):\n", __FILE__);
10
11      listDirectoriesAndFilesInRoot(L"D:\\Documents");
12      return 0;
13
14  }
15
PROBLEMS   DEBUG CONSOLE   TERMINAL   SERIAL MONITOR
OUTPUT                                                    Code                    DEBUG CONSOLE
[Running] cd "d:\Documents\Vs Code Projects\C language Coding Assignments\" &&
gcc 5RootDIrectoryNamesAdv.c -o 5RootDIrectoryNamesAdv && "d:\Documents\Vs Code
Projects\C language Coding Assignments\"5RootDIrectoryNamesAdv
Listing directories and files in D:\Documents (Current File Location:
5RootDIrectoryNamesAdv.c):
Directories:
Directory: Adobe
```

Retrieving code path advanced v1:

File Edit Selection View Go Run Terminal Help 5RootDIrectoryNamesAdv.c - Visual Studio Code

5RootDIrectoryNamesAdv.c

D: > Documents > Vs Code Projects > C language Coding > 5RootDIrectoryNamesAdv.c > main()

```c
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <wchar.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    // Get the current file path by extracting the folder path
    wchar_t currentFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentFilePath, FILENAME_MAX);

    // Find the last backslash to remove the file name and get the folder path
    wchar_t *lastBackslash = wcsrchr(currentFilePath, L'\\');
    if (lastBackslash != NULL)
    {
        *lastBackslash = L'\0';
    }

    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls):\n", currentFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}
```

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

OUTPUT Code DEBUG CONSOLE

```
[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc
5RootDIrectoryNamesAdv.c -o 5RootDIrectoryNamesAdv && "d:\Documents\Vs Code
Projects\C language Coding\"5RootDIrectoryNamesAdv
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding):
Directories:
Directory: Adobe
```

0 0 Ln 24, Col 14 Spaces: 4 UTF-8

Retrieving code path advanced v2:
The code was able to find:

1) Find the Current Code File Location of the parent directory:
   (Current Code File Location: d:\Documents\Vs Code Projects\C language Coding)
2) Find the path of the executable file contained:
   (*Current Code File: d:\Documents\Vs Code Projects\C language Coding\5RootDIrectoryNamesAdv.exe)*
3) Provide a listing of directories and files in D:\Documents, and it 1st lists the directories then files, in alphabetical order e.g.,
   i. Directories:
      Directory: Adobe
      Directory: Adobe Ps
      Directory: All My Custom App Presets
      Directory: Blender Projects
      Directory: Unity Projects

   ii. Files:
      File: desktop.ini
      File: DIr3.c
      File: DIr3.exe
      File: LCD Hello World .pdsprj

```c
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <wchar.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    // Get the current executable file path
    wchar_t currentExecutablePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentExecutablePath, FILENAME_MAX);

    // Find the last backslash to remove the file name and get the folder path
    wchar_t *lastBackslash = wcsrchr(currentExecutablePath, L'\\');
    if (lastBackslash != NULL)
    {
        *lastBackslash = L'\0';
    }

    // Get the current code file path
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    // Print the current code file location and the current code file
    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n(Current Code File: %ls):\n",
currentExecutablePath, currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);
```

```
    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            wprintf(L"File: %s\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```

File Edit Selection View Go Run Terminal Help 5RootDIrectoryName

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

OUTPUT Code

```
[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc
5RootDIrectoryNamesAdv.c -o 5RootDIrectoryNamesAdv && "d:\Documents\Vs Code
Projects\C language Coding\"5RootDIrectoryNamesAdv
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding)
(Current Code File: d:\Documents\Vs Code Projects\C language
Coding\5RootDIrectoryNamesAdv.exe):
Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUSim 8085
Directory: Hacker
Directory: Image-Line
Directory: Inkspace
```

⊗ 0 ⚠ 0

File Edit Selection View Go Run Terminal Help 5RootDIrector

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

OUTPUT Code

```
Directory: multisim
Directory: My InTouch Applications
Directory: National Instruments
Directory: Notepad
Directory: Nouva Projects
Directory: OneNote Notebooks
Directory: pdf
Directory: Project Backups
Directory: Proteus Projects
Directory: Scan2cad Projects
Directory: Siemens TIA Automation
Directory: Solid works
Directory: Sound Recordings
Directory: Unity
Directory: Unity Projects
Directory: VirtualDJ
Directory: Visual Studio 2022
Directory: VS 2022 Projects
Directory: Vs Code Projects
Files:
File: Default.rdp
File: desktop.ini
File: DIr3.c
File: DIr3.exe
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

[Done] exited with code=0 in 8.198 seconds
```

⊗ 0 ⚠ 0

The following code was able to add a bin folder: NB added in the int main() part only . however the fie executable path wasn't obtained correctly

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <wchar.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    // Get the current executable file path
    wchar_t currentExecutablePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentExecutablePath, FILENAME_MAX);

    // Find the last backslash to remove the file name and get the folder path
    wchar_t *lastBackslash = wcsrchr(currentExecutablePath, L'\\');
    if (lastBackslash != NULL)
    {
        *lastBackslash = L'\0';
    }

    // Create a copy of the current code file path before modifying it
    wchar_t currentCodeFilePath[FILENAME_MAX];
    wcscpy(currentCodeFilePath, currentExecutablePath);

    // Get the current code file path
    wchar_t currentCodeFileFullPath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFileFullPath, FILENAME_MAX);

    // Remove the executable file name from the current code file path
    wchar_t *lastBackslashCodeFile = wcsrchr(currentCodeFileFullPath, L'\\');
    if (lastBackslashCodeFile != NULL)
    {
        *lastBackslashCodeFile = L'\0';
    }

    // Create the "bin" directory in the current code file path
    wcscat(currentCodeFilePath, L"\\bin");
    CreateDirectoryW(currentCodeFilePath, NULL);

    // Get just the file name from the current code file path
    wchar_t *fileName = wcsrchr(currentCodeFileFullPath, L'\\');
    if (fileName != NULL)
    {
        fileName++; // Move past the backslash to the actual file name
    }
    else
    {
        fileName = L"Unknown"; // Fallback in case of any issue with the file name
    }

    // Print the current code file location and the current code file
    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n(Current Code File: %ls)\n(Current Code Bin
File Location: %ls):\n", currentExecutablePath, currentCodeFileFullPath, currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}
```

```
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding)
(Current Code File: d:\Documents\Vs Code Projects\C language Coding)
(Current Code Bin File Location: d:\Documents\Vs Code Projects\C language Coding\bin):
```

This was able to return the right output:

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <wchar.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    // Get the current executable file path
    wchar_t currentExecutablePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentExecutablePath, FILENAME_MAX);

    // Find the last backslash to remove the file name and get the folder path
    wchar_t *lastBackslash = wcsrchr(currentExecutablePath, L'\\');
    if (lastBackslash != NULL)
    {
        *lastBackslash = L'\0';
    }

    // Get just the file name from the current executable path
    wchar_t *fileName = wcsrchr(currentExecutablePath, L'\\');
    if (fileName != NULL)
    {
        fileName++; // Move past the backslash to the actual file name
    }
    else
    {
        fileName = L"Unknown"; // Fallback in case of any issue with the file name
    }

    // Create a copy of the current code file path before modifying it
    wchar_t currentCodeFilePath[FILENAME_MAX];
    wcscpy(currentCodeFilePath, currentExecutablePath);

    // Create the "bin" directory in the current code file path
    wcscat(currentCodeFilePath, L"\\bin");
    CreateDirectoryW(currentCodeFilePath, NULL);

    // Get the current code file name with .exe extension
    wchar_t currentCodeFileName[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFileName, FILENAME_MAX);

    // Print the current code file location and the current code file
    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n(Current Code File: %ls)\n(Current Code Bin
File Location: %ls):\n", currentExecutablePath, currentCodeFileName, currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
```

```c
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            wprintf(L"File: %s\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```

File Edit Selection View Go Run Terminal Help 6RootDIrectoryNamesAdv_CreateFolder.c - Visual Studio Code

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

OUTPUT Code DEBUG CONSOLE

Filter (e.g. text, !exclude)

```
Directory: Multisim
Directory: My InTouch Applications
Directory: National Instruments
Directory: Notepad
Directory: Nouva Projects
Directory: OneNote Notebooks
Directory: pdf
Directory: Project Backups
Directory: Proteus Projects
Directory: Scan2cad Projects
Directory: Siemens TIA Automation
Directory: Solid works
Directory: Sound Recordings
Directory: Unity
Directory: Unity Projects
Directory: VirtualDJ
Directory: Visual Studio 2022
Directory: VS 2022 Projects
Directory: Vs Code Projects
Files:
File: Default.rdp
File: desktop.ini
File: DIr3.c
File: DIr3.exe
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

[Done] exited with code=0 in 8.999 seconds
```

⊗ 0 ⚠ 0 Ln 91, Col 1 (3075 selected) Spaces: 4 UTF-8 CRLF {} C

File Edit Selection View Go Run Terminal Help 6RootDIrectoryNamesAdv_CreateFolder.c - Visual Studio Code

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

OUTPUT Code DEBUG CONSOLE

Filter (e.g. text, !exclude)

```
[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc 6RootDIrectoryNamesAdv_CreateFolder.c -o
6RootDIrectoryNamesAdv_CreateFolder && "d:\Documents\Vs Code Projects\C language Coding\"6RootDIrectoryNamesAdv_CreateFolder
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding)
(Current Code File: d:\Documents\Vs Code Projects\C language Coding\6RootDIrectoryNamesAdv_CreateFolder.exe)
(Current Code Bin File Location: d:\Documents\Vs Code Projects\C language Coding\bin):
Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUSim 8085
Directory: Hacker
Directory: Image-Line
Directory: Inkspace
Directory: Kicad
```

⊗ 0 ⚠ 0 Ln 91, Col 1 (3075 selected) Spaces: 4 UTF-8 CRLF {} C

**The following code didn't create a bin folder and a 1.txt file inside it. But it ran successfully after multiple tries. It also removed some functionalities

```
#include <stdio.h>
#include <windows.h>
#include <string.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\0';

    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n", currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        wprintf(L"Error finding directories and files in: %ls\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %ls\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            wprintf(L"File: %ls\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);

    // Create a new directory called "bin" in the current code file location
    wchar_t binDirectory[FILENAME_MAX];
    _snwprintf(binDirectory, FILENAME_MAX, L"%ls\\bin", path);
    CreateDirectoryW(binDirectory, NULL);

    // Create and log to a text file in the "bin" directory
    wchar_t logFileName[FILENAME_MAX];
    int suffix = 1;
    wchar_t suffixString[10];
    _snwprintf(suffixString, 10, L"%d", suffix);
    _snwprintf(logFileName, FILENAME_MAX, L"%ls\\log_%ls.txt", path, suffixString);

    FILE *logFile = _wfopen(logFileName, L"w");
```

```c
    if (logFile)
    {
        fwprintf(logFile, L"This is a sample log entry.\n");
        fclose(logFile);
    }
}
```

C Code Request | Bard | Syntax Highlighter

chat.openai.com/c/7703556d-852b-40a4-a9fb-1ea82f6daf49

New chat

Today

Adobe

C Code Request

Yesterday

Arduino AI From Scratch

Electron Chirality Explained

Friend's Workplace Job Applic

Database Exception (HYT00)

Previous 7 Days

Force Transmission Analysis

4000 YHz Hypothetical Frequ

Love Song Chord Progression

Upgrade to Plus NEW

[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc 7RootDIrectoryNamesAdv_CreateFolder_File.c -o 7RootDIrectoryNamesAdv_CreateFolder_File && "d:\Documents\Vs Code Projects\C language Coding\"7RootDIrectoryNamesAdv_CreateFolder_File
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding\)
Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: bin
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud

Regenerate response

Send a message

Battery status: 98% remaining

Today

Adobe

C Code Request

Yesterday

Arduino AI From Scratch

Electron Chirality Explained

Friend's Workplace Job Applic

Database Exception (HYT00)

Previous 7 Days

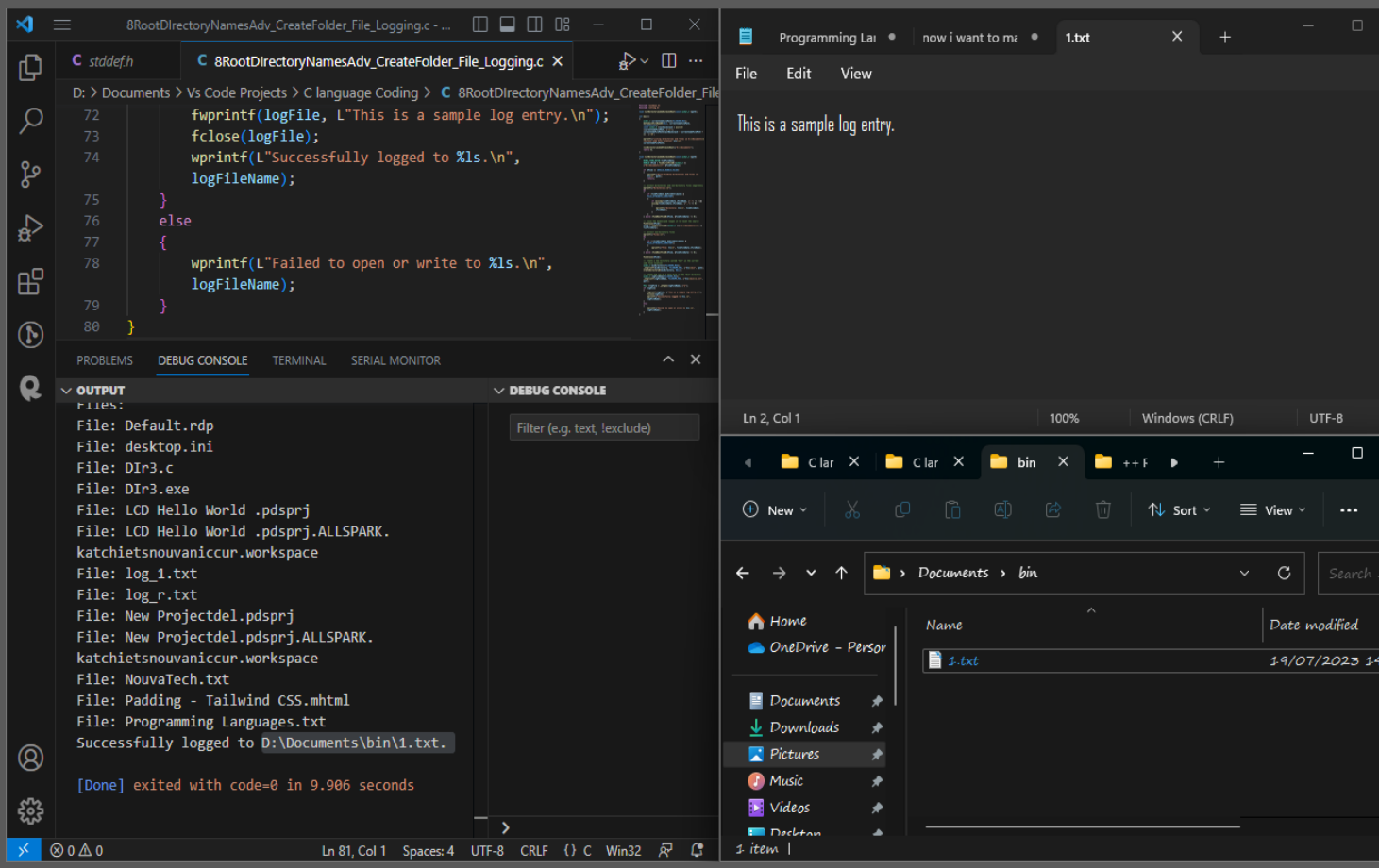
it didnt create a new directory bin such that a path D:\Documents\Vs Code Projects\C language Coding\bin exists and second if it had created bin folder, there woud be a text file inside it which the log would have appeared. i created a bin folder and rerun the code but there was no text file in it.

so let us solve the problem stepwise. since i already have created a bin folder such that the path D:\Documents\Vs Code Projects\C language Coding\bin exists, i have created a file inside it such that the path D:\Documents\Vs Code Projects\C language Coding\bin\1.txt exists. so your task is simple.
i need the result in the output to be logged into the 1.txt file

The following code was able to write into D:\Documents\bin\1.txt. the text This is a sample log entry

```
#include <stdio.h>
#include <windows.h>
#include <string.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\0';

    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n", currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        wprintf(L"Error finding directories and files in: %ls\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %ls\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            wprintf(L"File: %ls\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);

    // Create a new directory called "bin" in the current code file location
    wchar_t binDirectory[FILENAME_MAX];
    _snwprintf(binDirectory, FILENAME_MAX, L"%ls\\bin", path);
    CreateDirectoryW(binDirectory, NULL);

    // Create and log to a text file in the "bin" directory
    wchar_t logFileName[FILENAME_MAX];
    _snwprintf(logFileName, FILENAME_MAX, L"%ls\\bin\\1.txt", path);

    FILE *logFile = _wfopen(logFileName, L"w");
    if (logFile)
    {
        fwprintf(logFile, L"This is a sample log entry.\n");
        fclose(logFile);
        wprintf(L"Successfully logged to %ls.\n", logFileName);
```

```c
    }
    else
    {
        wprintf(L"Failed to open or write to %ls.\n", logFileName);
    }
}
```

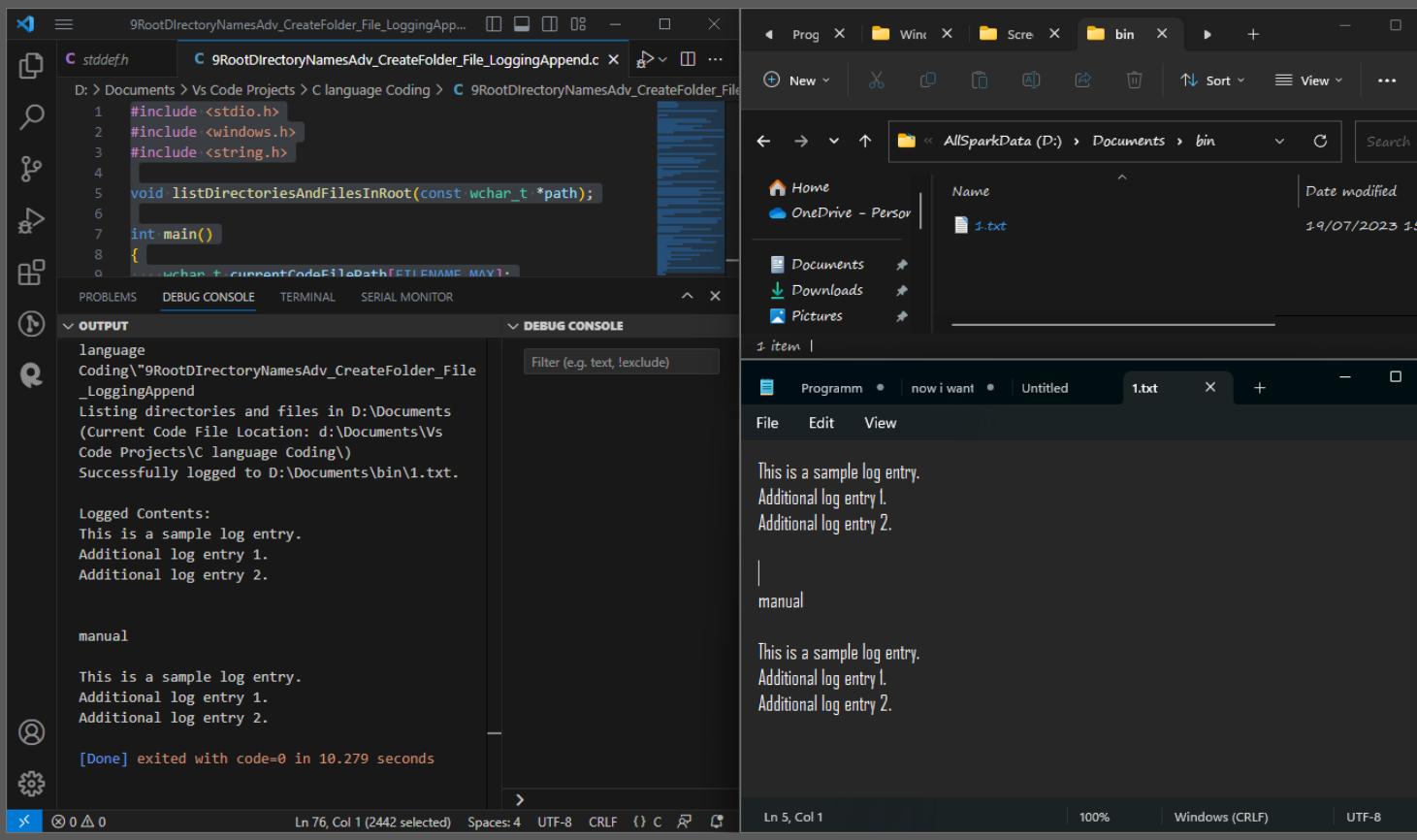


The code below is a symbol of advancement. It was able to:

1. **Create a new folder in the path D:\Documents in case it was non existent**
2. **Create a new file 1.txt incase it was non existent**
3. **Log in data into the text file.**
4. **Incase there was data in the log file it was able to append additional data into the log file.**

```
#include <stdio.h>
#include <windows.h>
#include <string.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\0';

    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n",
currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        wprintf(L"Error finding directories and files in: %ls\n", path);
        return;
    }

    // ... (Code to list directories and files remains unchanged)

    // Create the "bin" directory if it doesn't exist
    wchar_t binDirectory[FILENAME_MAX];
    _snwprintf(binDirectory, FILENAME_MAX, L"%ls\\bin", path);
    CreateDirectoryW(binDirectory, NULL);

    // Create and log to a text file in the "bin" directory
    wchar_t logFileName[FILENAME_MAX];
    _snwprintf(logFileName, FILENAME_MAX, L"%ls\\bin\\1.txt", path);

    FILE *logFile = _wfopen(logFileName, L"a"); // Open in "append" mode instead of "write" mode
    if (logFile)
    {
        fwprintf(logFile, L"This is a sample log entry.\n");
        fwprintf(logFile, L"Additional log entry 1.\n");
        fwprintf(logFile, L"Additional log entry 2.\n");

        fclose(logFile);
        wprintf(L"Successfully logged to %ls.\n", logFileName);
```

```c
        // Read and print the contents of the log file to the console
        wprintf(L"\nLogged Contents:\n");
        FILE *readLogFile = _wfopen(logFileName, L"r");
        if (readLogFile)
        {
            wchar_t buffer[512];
            while (fgetws(buffer, 512, readLogFile))
            {
                wprintf(L"%ls", buffer);
            }
            fclose(readLogFile);
        }
        else
        {
            wprintf(L"Failed to read %ls.\n", logFileName);
        }
    }
    else
    {
        wprintf(L"Failed to open or write to %ls.\n", logFileName);
    }

    // ... (The rest of the code remains unchanged)
}
```



Advanced Desktop Application

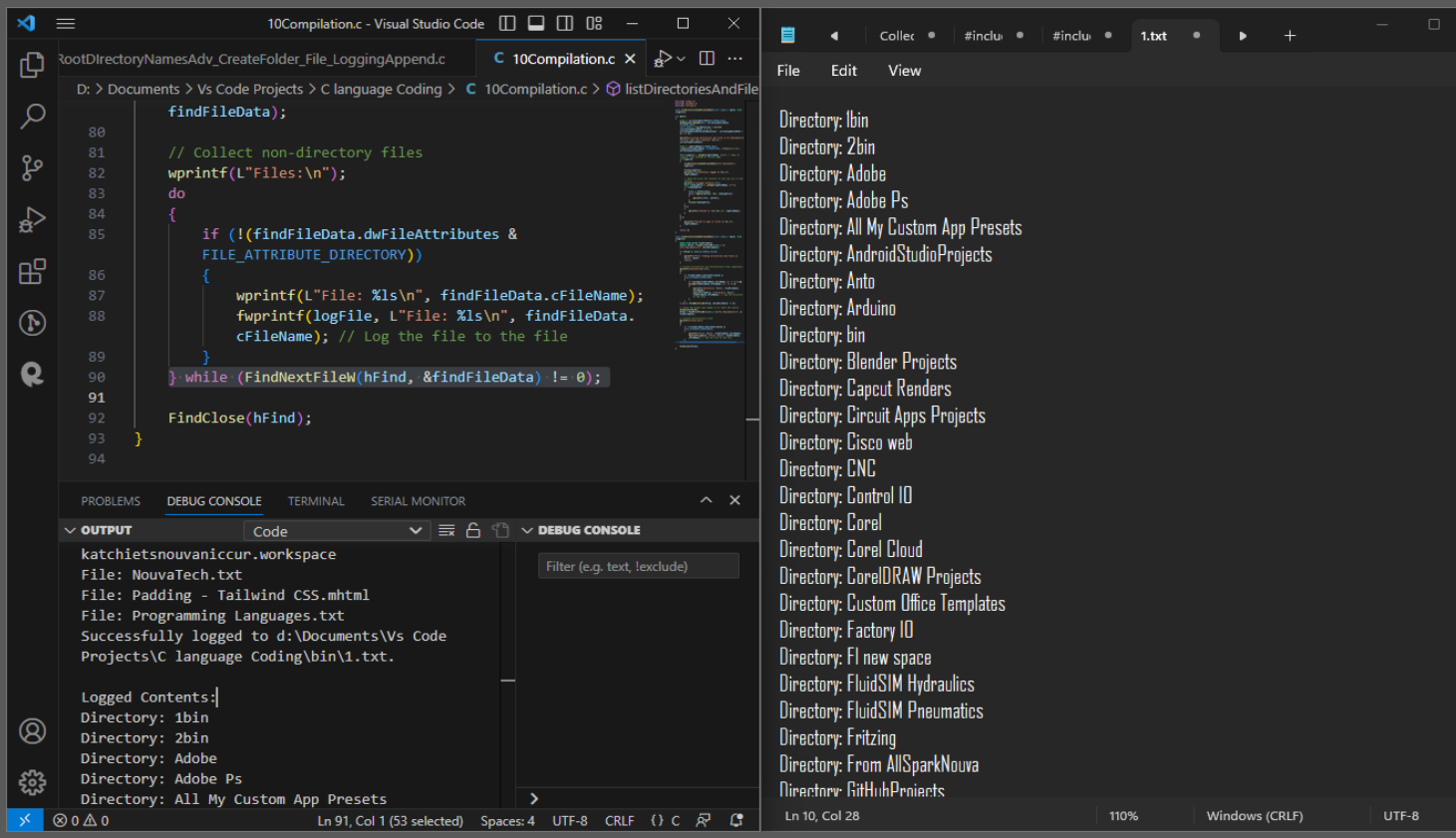
The code below is a symbol of Super advancement. It was able to:

1. **Check if the bin folder exists in the root of the installation folder**
2. **If not, it will create it before attempting to open the log file (This should prevent the "Failed to open or write to" error from occurring when the folder does not exist.)**
3. **Create a new file 1.txt in case it was nonexistent.**
4. **Write the root contents of path D:\Documents into the log file in installation folder\bin\1.txt**
5. **If the log file already exists, it will append the data to the already existing contents.**
6. **I successfully copied it to another drive and could do step 1 through to 5 without problem**

```c
#include <stdio.h>
#include <windows.h>
#include <string.h>

// Function prototype declaration
void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);

int main()
{
    // Variable to store the path of the current code file
    wchar_t currentCodeFilePath[FILENAME_MAX];
    // Get the path of the current executable (the code file)
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    // Extract the directory path by removing the file name from the path
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\0';

    // Print the current directory and the code file's location
    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n", currentCodeFilePath);

    // Variable to store the log file name
    wchar_t logFileName[FILENAME_MAX];
    // Create the log file name using the current directory path
    _snwprintf(logFileName, FILENAME_MAX, L"%lsbin\\1.txt", currentCodeFilePath);

    // Check if the 'bin' folder exists, if not, create it
    wchar_t binFolderPath[FILENAME_MAX];
    _snwprintf(binFolderPath, FILENAME_MAX, L"%lsbin", currentCodeFilePath);
    if (!CreateDirectoryW(binFolderPath, NULL))
    {
        DWORD error = GetLastError();
        if (error != ERROR_ALREADY_EXISTS)
        {
            // Print an error message if the folder creation fails
            wprintf(L"Failed to create 'bin' folder: %ls\n", binFolderPath);
            return 1; // Return with an error code (1) to indicate failure
        }
    }

    // Open the log file in "append" mode (add data to the existing file)
    FILE *logFile = _wfopen(logFileName, L"a");
    if (logFile)
    {
        // Call the function to list directories and files in the specified path
        listDirectoriesAndFilesInRoot(L"D:\\Documents", logFile);

        // Close the log file after writing the data
        fclose(logFile);
```

```
        // Print a success message with the log file name
        wprintf(L"Successfully logged to %ls.\n", logFileName);

        // Read and print the contents of the log file to the console
        wprintf(L"\nLogged Contents:\n");
        FILE *readLogFile = _wfopen(logFileName, L"r");
        if (readLogFile)
        {
            wchar_t buffer[512];
            while (fgetws(buffer, 512, readLogFile))
            {
                wprintf(L"%ls", buffer);
            }
            fclose(readLogFile);
        }
        else
        {
            // Print an error message if reading the log file fails
            wprintf(L"Failed to read %ls.\n", logFileName);
        }
    }
    else
    {
        // Print an error message if opening or writing to the log file fails
        wprintf(L"Failed to open or write to %ls.\n", logFileName);
    }

    return 0; // Return with a success code (0) to indicate successful execution
}

// Function definition to list directories and files in the specified path
void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
{
    // Variables for handling file enumeration in the specified path
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Check if file enumeration is successful or not
    if (hFind == INVALID_HANDLE_VALUE)
    {
        // Print an error message if enumeration fails
        wprintf(L"Error finding directories and files in: %ls\n", path);
        return; // Return from the function
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\n");
    do
    {
        // Check if the current item is a directory
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            // Exclude '.' and '..' directories from the listing
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..")
!= 0)
            {
                // Print the directory name to the console
```

```c
                wprintf(L"Directory: %ls\n", findFileData.cFileName);
                // Log the directory name to the log file
                fwprintf(logFile, L"Directory: %ls\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
        // Check if the current item is not a directory (i.e., a file)
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            // Print the file name to the console
            wprintf(L"File: %ls\n", findFileData.cFileName);
            // Log the file name to the log file
            fwprintf(logFile, L"File: %ls\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle to release resources
    FindClose(hFind);
}

```



To change the path you want to look into (e.g., D:\dukto instead of D:\Documents), you need to modify two parts of the code:

1. In the `main()` function:

```
int main()
{
    // ...
    // Replace the path "D:\\Documents" with "D:\\dukto"
    listDirectoriesAndFilesInRoot(L"D:\\dukto", logFile);
    // ...
}

```

In the `listDirectoriesAndFilesInRoot` function:

```
void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
{
// ...
// Replace the path "D:\\Documents\\*" with the new path "D:\\dukto\\*"
HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\dukto\\*", &findFileData);
// ...
}
```

After making these changes, the program will list directories and files in the specified path (D:\dukto) instead of D:\Documents.

Remember to ensure that the specified path (D:\dukto) exists and contains the files and directories you want to list; otherwise, the program might not produce the expected results.

I wanted to access the default folders. Now suppose that normally in every computer, by default there is a downloads, document, videos, pictures, music and desktop folders. In most operating systems, especially Windows, macOS, and Linux distributions, there are standard default folders commonly found in the user's home directory. These folders are created by the system to organize specific types of files and are commonly known as "special folders" or "user folders."

The standard default folders you mentioned are commonly found in many systems:

1. **Downloads: This folder is intended to store downloaded files from the internet or other sources.**
2. **Documents: It's used to store user documents and other text-based files.**
3. **Videos: This folder is meant to hold video files.**
4. **Pictures: It's used to store image files and photos.**
5. **Music: This folder is meant to hold audio files, like music tracks.**
6. **Desktop: It's used to store files and shortcuts that are displayed on the user's desktop.**

The actual names of these folders might vary slightly depending on the operating system and its language settings. For example, on some systems, "Documents" might be called "My Documents" or "Documents" in the native language.

These folders are intended to help users organize their files conveniently, and most applications typically save and access user-specific files from these standard folders to promote a consistent user experience.

It's important to note that while these default folders are commonly present in most systems, users can still create, rename, or delete folders to suit their preferences or organizational needs. Additionally, the availability and names of these folders can differ in various operating systems and versions.

To access the standard default folders without using hardcoded paths, you can use platform-specific functions that provide the paths to these special folders programmatically. In Windows, you can use the SHGetFolderPath function to get the paths to special folders. On macOS, you can use NSSearchPathForDirectoriesInDomains, and on Linux, you can use XDG Base Directory Specification.

Here's how you can modify the code to access the "Downloads" folder on Windows using SHGetFolderPath

Added the #include <shlobj.h> header to include the necessary declarations for using the SHGetFolderPath function.

Modified the main() function to use SHGetFolderPath to get the path of the "Downloads" folder:

The SHGetFolderPath function retrieves the path of a special folder identified by a CSIDL (Constant Special Item ID List) value. In this case, CSIDL_PERSONAL is used, which represents the "My Documents" folder on Windows, where the "Downloads" folder is often located.

After retrieving the path using SHGetFolderPath, the code appends "\Downloads" to the path to form the full path to the "Downloads" folder. The rest of the code then uses this dynamically obtained path to list directories and files in the "Downloads" folder.

Using this approach allows the code to access the "Downloads" folder without relying on hardcoded paths, making it more flexible and portable across different systems and users.

```
// ... Your existing function implementation remains unchanged ...
```

From code number 57

```
#include <stdio.h>
#include <windows.h>
#include <shlobj.h> // Include the header for SHGetFolderPath

void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);

int main()
{
    // Variable to store the path of the Downloads folder
    wchar_t downloadsFolderPath[MAX_PATH];

    // Get the path of the Downloads folder using SHGetFolderPath
    if (SUCCEEDED(SHGetFolderPathW(NULL, CSIDL_PERSONAL, NULL, 0, downloadsFolderPath)))
    {
        // Append "\\Downloads" to the path to get the full Downloads folder path
        wcscat(downloadsFolderPath, L"\\Downloads");

```

```
        // Open the log file in "append" mode (add data to the existing file)
        FILE *logFile = _wfopen(L"log.txt", L"a");
        if (logFile)
        {
            // Call the function to list directories and files in the specified path
            listDirectoriesAndFilesInRoot(downloadsFolderPath, logFile);

            fclose(logFile);
            wprintf(L"Successfully logged to log.txt.\n");

            // Read and print the contents of the log file to the console
            wprintf(L"\nLogged Contents:\n");
            FILE *readLogFile = _wfopen(L"log.txt", L"r");
            if (readLogFile)
            {
                wchar_t buffer[512];
                while (fgetws(buffer, 512, readLogFile))
                {
                    wprintf(L"%ls", buffer);
                }
                fclose(readLogFile);
            }
            else
            {
                wprintf(L"Failed to read log.txt.\n");
            }
        }
        else
        {
            wprintf(L"Failed to open or write to log.txt.\n");
        }
    }
    else
    {
        wprintf(L"Failed to get the path of the Downloads folder.\n");
    }
    return 0; // Return with a success code (0) to indicate successful execution
}

// ... Your existing function implementation remains unchanged ...

// Function definition to list directories and files in the specified path
void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
{
    // Variables for handling file enumeration in the specified path
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);

    // Check if file enumeration is successful or not
    if (hFind == INVALID_HANDLE_VALUE)
    {
        // Print an error message if enumeration fails
        wprintf(L"Error finding directories and files in: %ls\n", path);
        return; // Return from the function
    }

    // Collect directories and non-directory files separately
```

```
    wprintf(L"Directories:\n");
    do
    {
        // Check if the current item is a directory
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            // Exclude '.' and '..' directories from the listing
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                // Print the directory name to the console
                wprintf(L"Directory: %ls\n", findFileData.cFileName);
                // Log the directory name to the log file
                fwprintf(logFile, L"Directory: %ls\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
                // Check if the current item is not a directory (i.e., a file)
                if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
                {
                    // Print the file name to the console
                    wprintf(L"File: %ls\n", findFileData.cFileName);
                    // Log the file name to the log file
                    fwprintf(logFile, L"File: %ls\n", findFileData.cFileName);
                }
            } while (FindNextFileW(hFind, &findFileData) != 0);

            // Close the handle to release resources
            FindClose(hFind);
        }

```

14FileSniffer_for_Defult_Directories.c - Visual Studio Code

14FileSniffer_for_Defult_Directories.c

D: > Documents > Vs Code Projects > C language Coding > 14FileSniffer_for_Defult_Directories.c > ...

```c
#include <stdio.h>
#include <windows.h>
#include <shlobj.h> // Include the header for SHGetFolderPath

void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);

int main()
{
    // Variable to store the path of the Downloads folder
    wchar_t downloadsFolderPath[MAX_PATH];

    // Get the path of the Downloads folder using SHGetFolderPath
    if (SUCCEEDED(SHGetFolderPathW(NULL, CSIDL_PERSONAL, NULL, 0
```

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

OUTPUT Code

```
File: Xklusiv Dancers Tun up the Scheme a lIttle @ 2011.mp4
File: XXXTENTACION  - Royalty (Official Video) (feat. Ky-Mani
Marley, Stefflon Don & Vybz Kartel).mp4
File: XXXTENTACION & Lil Pump - Arms Around You feat. Maluma &
Swae Lee [Official Music Video].mp4
File: XXXTENTACION & Lil Pump - Arms Around You feat. Maluma &
Swae Lee [Official Music Video]_2.mkv
File: Yeshua.mp4
File: Young Thug - Hoe Tendencies (unreleased).mp4
File: Young Thug - Hoe Tendencies (unreleased)_2.mp4
File: Young Thug - Out The Ghetto [432hz].mp4
File: Young Thug Type Beat X Gunna Type Beat 2023 - Shooting Star
- Free Type Beat - Rap_Trap Instrumental.mp4
File: YPB Song 2018.mp4
File: [FREE FOR PROFIT] NATTY X MANDO GTAtv DRILL TYPE BEAT -
NATTORIUS APOLLO 2023.mp4
File: _Getintopc.com_Adobe_Lightroom_Classic_2023.mp4
File: _Getintopc.com_Adobe_Photoshop_Lightroom_CC_6.8_Portable.mp4
File: _Getintopc.com_Scan2CAD_2022.mp4
File:  How to become a Web3 Blockchain Developer- Ft. Peter
Holzer.vtt
File:  Masicka & Bounty Killer, Vybz Kartel .mp4
File:  (via@@Dennis Teo) #shorts.mp4

[Done] exited with code=0 in 20.688 seconds
```

DEBUG CONSOLE

Filter (e.g. text, !exclude)

0 0 Ln 56, Col 1 (66 selected) Spaces: 4 UTF-8 CRLF {} C Win32

Collec #inclu #inclu 1.txt

File Edit View

```
Directory: 3D
Directory: All DBMS and SQL
Directory: And St
Directory: ar projects
Directory: Arduino
Directory: Azure
Directory: bin
Directory: Calender
Directory: Cisco
Directory: Compressed
Directory: Connecting components in Proteus
Directory: DApp
Directory: Documents
Directory: From pc
Directory: G code
Directory: IDM Cache
Directory: Images
Directory: IR and LCD Simulation Proteus
Directory: MicrosoftWindows.Client.CBS_cw5nlh2txyewy!InputApp
Directory: Motor driver
Directory: Movies
Directory: Music
Directory: nF
Directory: Pictures
Directory: Programs
Directory: Robotics
```

Ln 1, Col 1 110% Windows (CRLF) ANSI

The following code is a template of default folders iteration

```
#include <stdio.h>
#include <windows.h>
#include <shlobj.h> // Include the header for SHGetFolderPath

void processFolder(const wchar_t *folderName);

int main()
{
    // Define an array of folder names
    const wchar_t *defaultFolders[] = {
        L"Downloads",
        L"Documents",
        L"Videos",
        L"Pictures",
        L"Music",
        L"Desktop"
    };

    // Iterate through the array and process each folder
    for (int i = 0; i < sizeof(defaultFolders) / sizeof(defaultFolders[0]); i++)
    {
        processFolder(defaultFolders[i]);
    }

    return 0;
}

// Function to process the contents of a folder
void processFolder(const wchar_t *folderName)
{
    // Variable to store the path of the folder
    wchar_t folderPath[MAX_PATH];

    // Get the path of the specified folder using SHGetFolderPath
    if (SUCCEEDED(SHGetFolderPathW(NULL, CSIDL_PERSONAL, NULL, 0, folderPath)))
    {
        // Append the folder name to the path to get the full folder path
        wcscat(folderPath, L"\\");
        wcscat(folderPath, folderName);

        // Perform operations on the folder here...
        // For example, you can list directories and files in the folder, or do any other processing.

        // For demonstration purposes, let's just print the folder path to the console
        wprintf(L"Folder: %ls\n", folderPath);
    }
    else
    {
        wprintf(L"Failed to get the path of the %ls folder.\n", folderName);
    }
}
```

15_Iterate_through_default_Windows_folders.c - Visual Stu...

Directories.c | 15_Iterate_through_default_Windows_folders.c | 16Master_Default_Folder_sniffer.c

D: > Documents > Vs Code Projects > C language Coding > 15_Iterate_through_default_Windows_folders.c > processFolder(

```c
#include <stdio.h>
#include <windows.h>
#include <shlobj.h> // Include the header for SHGetFolderPath

void processFolder(const wchar_t *folderName);

int main()
{
    // Define an array of folder names
    const wchar_t *defaultFolders[] = {
        L"Downloads",
        L"Documents",
        L"Videos",
        L"Pictures",
        L"Music",
        L"Desktop"
    };

```

PROBLEMS | DEBUG CONSOLE | TERMINAL | SERIAL MONITOR

OUTPUT

```
[Running] cd "d:\Documents\Vs Code
Projects\C language Coding\" && gcc
15_Iterate_through_default_Windows_folders.c
-o
15_Iterate_through_default_Windows_folders &&
 "d:\Documents\Vs Code Projects\C language
Coding\"15_Iterate_through_default_Windows_fo
lders
Folder: D:\Documents\Downloads
Folder: D:\Documents\Documents
Folder: D:\Documents\Videos
Folder: D:\Documents\Pictures
Folder: D:\Documents\Music
Folder: D:\Documents\Desktop

[Done] exited with code=0 in 7.669 seconds
```

DEBUG CONSOLE

Filter (e.g. text, !exclude)

0 0 | Ln 39, Col 40 | Spaces: 4 | UTF-8 | CRLF | {} C | Win32

bin | Home | AllSparkData

New | Sort | View

This PC > AllSparkData (D:) >

Search

- Home
- OneDrive - Personal
- Documents
- Downloads
- Pictures
- Music
- Videos
- Desktop
- Msc
- Screenshots
- C language Coding
- progress bin
- Learning C Reports
- Creative Cloud Files
- This PC
- Local Disk (C:)
- AllSparkData (D:)
- Network

Name	Date modified	Type
$AV_ASW	12/06/2023 10:...	File f
+GForce	19/07/2023 07:...	File f
avast! sandbox	19/07/2023 09:...	File f
Desktop	19/07/2023 11:...	File f
Documents	19/07/2023 18:...	File f
Downloads	19/07/2023 17:...	File f
Dukto	26/05/2023 21:...	File f
Msc	11/07/2023 12:...	File f
Music	19/07/2023 08:...	File f
Pictures	18/07/2023 09:...	File f
University Days	01/07/2023 11:...	File f
Videos	19/07/2023 17:...	File f
IMG-20230223-WA0001.jpg	31/05/2023 20:...	JPG
Network - Shortcut	25/06/2023 23:...	Shor
NouvaTech	16/04/2023 19:...	File

15 items

The above code had the following limitations:

The code you provided seems to be a C program that lists directories and files in specific folders on your computer and logs the results to a file. The folders it lists are the default ones like Downloads, Documents, Videos, Pictures, Music, and Desktop. However, there are a few issues in the code that need to be addressed.

Hardcoded Path: The code currently lists the directories and files in the "D:\Documents" folder. To make it more flexible and work on any system, you can replace the hardcoded path with a dynamic path using the SHGetFolderPath function.

Folder Enumeration: The listDirectoriesAndFilesInRoot function currently uses the "D:\Documents" folder directly for enumeration. Instead, you should use the path parameter that's passed to the function.

Incorrect Folder Enumeration: The listDirectoriesAndFilesInRoot function doesn't properly use the provided path for enumeration. It uses a hardcoded path instead. To fix this, you need to replace (wchar_t *)L"D:\\Documents*" with (wchar_t *)path in both places.

Incorrect Log Folder Path: In the processDefaultFolders function, when calling listDirectoriesAndFilesInRoot, it passes the folder names as paths directly. Instead, it should use the dynamic path obtained from SHGetFolderPath for each default folder.

With these modifications, the code should now correctly list the directories and files in the default folders and log the results to the specified log file. Additionally, it uses the SHGetFolderPath function to dynamically obtain the path of the "Documents" folder, making the code more adaptable to different systems

The listDirectoriesAndFilesInRoot function is not correctly using the provided path for enumeration, and it's always using the "D:\Documents" folder for listing, which leads to the incorrect log.

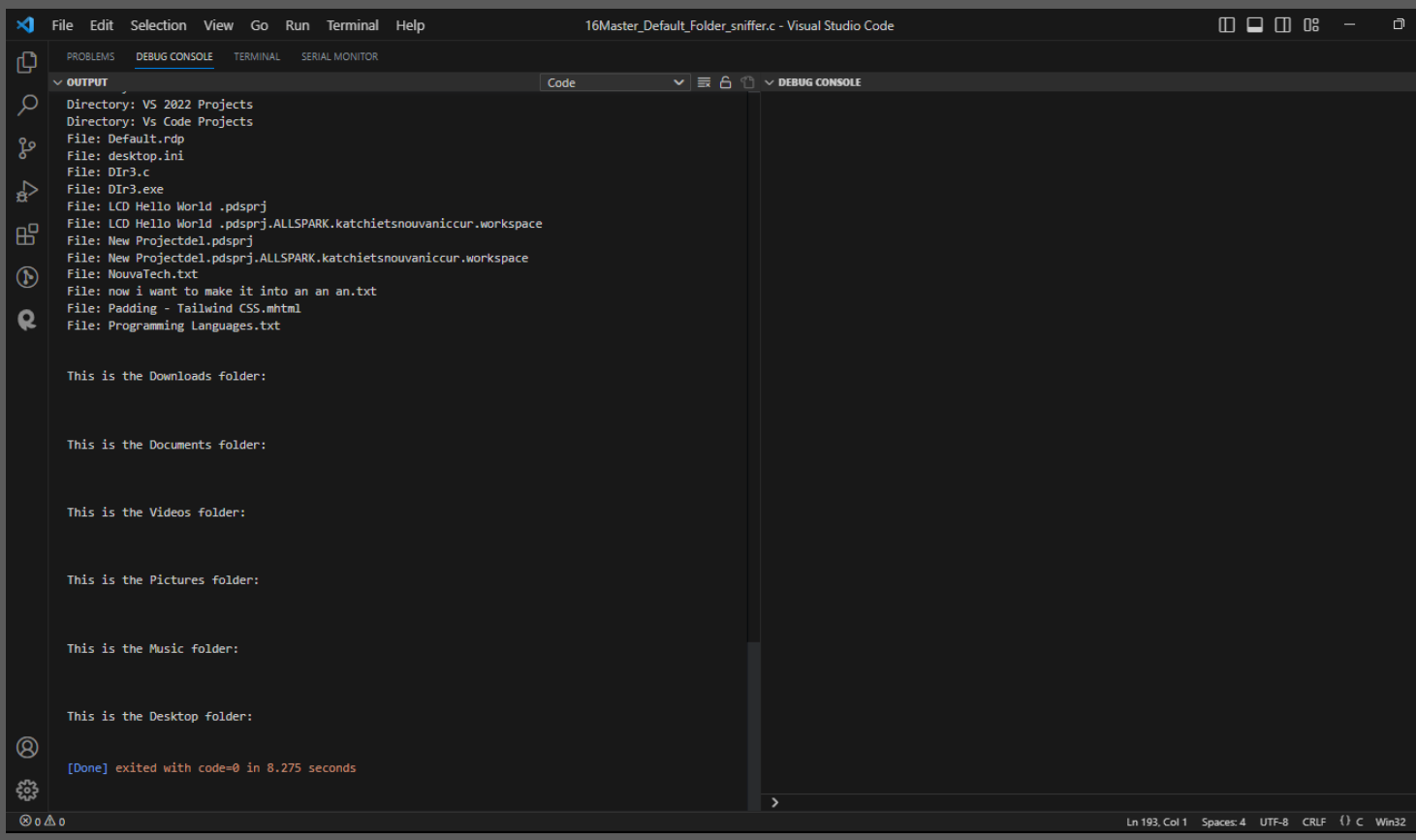
To fix this, we need to replace the hardcoded path (wchar_t *)L"D:\\Documents*" with the path parameter in the listDirectoriesAndFilesInRoot function.

still an issue with the code when it comes to listing the contents of the default folders like Downloads, Documents, Videos, etc. The code is not correctly listing the contents of those folders, and it shows "Error finding directories and files" for each of them.

The problem lies in the processDefaultFolders function, where we are not correctly passing the paths of the default folders to the listDirectoriesAndFilesInRoot function. We need to provide the correct paths to the listDirectoriesAndFilesInRoot function.

To fix this, we need to use the SHGetFolderPath function to obtain the paths of the default folders dynamically

With this modification, the processDefaultFolders function should correctly list the contents of each default folder and log them to the file.



t there's still an issue with the code when listing the contents of the default folders. The issue lies in the way we are using the SHGetFolderPath function to obtain the paths of the default folders.

Instead of using SHGetFolderPath for each default folder, we can directly access the default folders using the corresponding CSIDLs (constants). We don't need to use SHGetFolderPath for these default folders because they have predefined CSIDL values.

Let's modify the processDefaultFolders function to directly access the default folders using their CSIDL values:

To include the Downloads folder in the output, you can add the CSIDL value for the Downloads folder (CSIDL_PERSONAL) to the defaultFolderCSIDLs array in the processDefaultFolders function.

The following code printed for all the default folders except for the downloads folder:
It displayed for :

1. **Desktop folder**
2. **Documents folder**
3. **Videos folder**
4. **Pictures folder**
5. **Music folder**
6. **Common Desktop folder**

It didn't display correctly for:

1. **Downloads folder**

```c
#include <stdio.h>
#include <windows.h>
#include <shlobj.h> // Include the header for SHGetFolderPath

// Function prototypes
void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);
void processDefaultFolders(FILE *logFile);

int main()
{
    // Variable to store the path of the current code file
    wchar_t currentCodeFilePath[FILENAME_MAX];
    // Get the path of the current executable (the code file)
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    // Extract the directory path by removing the file name from the path
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\0';

    // Get the path of the 'Documents' folder using SHGetFolderPath
    wchar_t documentsFolderPath[FILENAME_MAX];
    if (SHGetFolderPathW(NULL, CSIDL_PERSONAL, NULL, 0, documentsFolderPath) != S_OK)
    {
        // Print an error message if getting the folder path fails
        wprintf(L"Failed to get 'Documents' folder path.\n");
        return 1; // Return with an error code (1) to indicate failure
    }

    // Print the current directory and the code file's location
    wprintf(L"Listing directories and files in %ls\n(Current Code File Location: %ls)\n", documentsFolderPath, currentCodeFilePath);

    // Variable to store the log file name
    wchar_t logFileName[FILENAME_MAX];
    // Create the log file name using the current directory path
    _snwprintf(logFileName, FILENAME_MAX, L"%lsbin\\1.txt", currentCodeFilePath);

    // Check if the 'bin' folder exists, if not, create it
    wchar_t binFolderPath[FILENAME_MAX];
    _snwprintf(binFolderPath, FILENAME_MAX, L"%lsbin", currentCodeFilePath);
    if (!CreateDirectoryW(binFolderPath, NULL))
    {
        DWORD error = GetLastError();
        if (error != ERROR_ALREADY_EXISTS)
        {
            // Print an error message if the folder creation fails
            wprintf(L"Failed to create 'bin' folder: %ls\n", binFolderPath);
            return 1; // Return with an error code (1) to indicate failure
        }
```

```
    }

    // Open the log file in "append" mode (add data to the existing file)
    FILE *logFile = _wfopen(logFileName, L"a");
    if (logFile)
    {
        // Call the function to list directories and files in the specified path (Documents folder)
        listDirectoriesAndFilesInRoot(documentsFolderPath, logFile);

        // Call the new function to iterate through the default folders and append data to the log file
        processDefaultFolders(logFile);

        // Close the log file after writing the data
        fclose(logFile);
        // Print a success message with the log file name
        wprintf(L"Successfully logged to %ls.\n", logFileName);

        // Read and print the contents of the log file to the console
        wprintf(L"\nLogged Contents:\n");
        FILE *readLogFile = _wfopen(logFileName, L"r");
        if (readLogFile)
        {
            wchar_t buffer[512];
            while (fgetws(buffer, 512, readLogFile))
            {
                wprintf(L"%ls", buffer);
            }
            fclose(readLogFile);
        }
        else
        {
            // Print an error message if reading the log file fails
            wprintf(L"Failed to read %ls.\n", logFileName);
        }
    }
    else
    {
        // Print an error message if opening or writing to the log file fails
        wprintf(L"Failed to open or write to %ls.\n", logFileName);
    }

    return 0; // Return with a success code (0) to indicate successful execution
}

// Function definition to list directories and files in the specified path
void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
{
    // Variables for handling file enumeration in the specified path
    wchar_t searchPath[MAX_PATH];
    WIN32_FIND_DATAW findFileData;

    // Create the search pattern for the specified path
        _snwprintf(searchPath, MAX_PATH, L"%s\\*", path);

        // Find the first file in the specified path
        HANDLE hFind = FindFirstFileW(searchPath, &findFileData);
```

```

        // Check if file enumeration is successful or not
        if (hFind == INVALID_HANDLE_VALUE)
        {
            // Print an error message if enumeration fails
            wprintf(L"Error finding directories and files in: %ls\n", path);
            return; // Return from the function
        }

        // Collect directories and non-directory files separately
        wprintf(L"Directories:\n");
        do
        {
            // Check if the current item is a directory
            if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
            {
                // Exclude '.' and '..' directories from the listing
                if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
                {
                    // Print the directory name to the console
                    wprintf(L"Directory: %ls\n", findFileData.cFileName);
                    // Log the directory name to the log file
                    fwprintf(logFile, L"Directory: %ls\n", findFileData.cFileName);
                }
            }
        } while (FindNextFileW(hFind, &findFileData) != 0);

        // Close the handle to release resources
        FindClose(hFind);

        // Reopen the handle to find non-directory files
        hFind = FindFirstFileW(searchPath, &findFileData);

        // Collect non-directory files
        wprintf(L"Files:\n");
        do
        {
            // Check if the current item is not a directory (i.e., a file)
            if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
            {
                // Print the file name to the console
                wprintf(L"File: %ls\n", findFileData.cFileName);
                // Log the file name to the log file
                fwprintf(logFile, L"File: %ls\n", findFileData.cFileName);
            }
        } while (FindNextFileW(hFind, &findFileData) != 0);

        // Close the handle to release resources
        FindClose(hFind);
    }

    void processDefaultFolders(FILE *logFile)
    {
        // Define an array of folder CSIDLs (constants) along with their descriptions
        const int defaultFolderCSIDLs[] = {
```

```
159.                CSIDL_DESKTOP, CSIDL_PERSONAL, CSIDL_MYVIDEO, CSIDL_MYPICTURES, CSIDL_MYMUSIC,
    CSIDL_COMMON_DESKTOPDIRECTORY, 0x001A
160.            };
161.
162.            // Define the corresponding folder descriptions
163.            const wchar_t *defaultFolderDescriptions[] = {
164.                L"This is the Desktop folder:",
165.                L"This is the Documents folder:",
166.                L"This is the Videos folder:",
167.                L"This is the Pictures folder:",
168.                L"This is the Music folder:",
169.                L"This is the Common Desktop folder:",
170.                L"This is the Downloads folder:"
171.            };
172.
173.            // Iterate through the array and call listDirectoriesAndFilesInRoot for each folder
174.            for (int i = 0; i < sizeof(defaultFolderCSIDLs) / sizeof(defaultFolderCSIDLs[0]); i++)
175.            {
176.                // Get the path of the default folder using CSIDL
177.                wchar_t folderPath[MAX_PATH];
178.                if (SHGetFolderPathW(NULL, defaultFolderCSIDLs[i], NULL, 0, folderPath) == S_OK)
179.                {
180.                    // Add the folder description to the log file
181.                    fwprintf(logFile, L"\n\n%s\n\n", defaultFolderDescriptions[i]);
182.
183.                    // Call listDirectoriesAndFilesInRoot for each default folder
184.                    listDirectoriesAndFilesInRoot(folderPath, logFile);
185.                }
186.                else
187.                {
188.                    // Print an error message if getting the folder path fails
189.                    wprintf(L"Failed to get default folder path for CSIDL %d.\n",
    defaultFolderCSIDLs[i]);
190.                }
191.            }
192.        }
```

16Master_Default_Folder_sniffer.c - Visual Studio Code

irectories.c | 15_Iterate_through_default_Windows_folders.c | 16Master_Default_Folder_sniffer.c

D: > Documents > Vs Code Projects > C language Coding > 16Master_Default_Folder_sniffer.c > processDefaultFolders(FILE

```c
{
    // Define an array of folder CSIDLs (constants) along with their descriptions
    const int defaultFolderCSIDLs[] = {
        CSIDL_DESKTOP, CSIDL_PERSONAL, CSIDL_MYVIDEO, CSIDL_MYPICTURES,
        CSIDL_MYMUSIC, CSIDL_COMMON_DESKTOPDIRECTORY, 0x001A
    };

    // Define the corresponding folder descriptions
    const wchar_t *defaultFolderDescriptions[] = {
        L"This is the Desktop folder:",
        L"This is the Documents folder:",
        L"This is the Videos folder:",
        L"This is the Pictures folder:",
        L"This is the Music folder:",
        L"This is the Common Desktop folder:",
        L"This is the Downloads folder:"
    };

    // Iterate through the array and call listDirectoriesAndFilesInRoot for each folder
    for (int i = 0; i < sizeof(defaultFolderCSIDLs) / sizeof(defaultFolderCSIDLs[0]); i++)
    {
```

PROBLEMS | DEBUG CONSOLE | TERMINAL | SERIAL MONITOR

OUTPUT

```
Directory: Unity
Directory: UnityHub
Directory: Visual Studio Setup
Directory: vlc
Directory: WindowsActiveServices
Directory: WindowsServices
Directory: WinRAR
File: s2csettings.s1
File: s2csettings.s2
File: s2csettings.s3
File: s2csettings.s4

[Done] exited with code=0 in 9.721 seconds
```

DEBUG CONSOLE

Filter (e.g. text, !exclude)

0 0 | Ln 164, Col 1 (345 selected) | Spaces: 4 | UTF-8 | CRLF | {} C | Win32

bin | AllSp | ++ F | Dow

New | Sort | View

C language Coding > bin

Search

Home
OneDrive - Persor
Documents
Downloads

Name	Date modified	Type	Size
1.txt	19/07/2023 23:30	Text Docum...	1

1 item | 1 item selected 9.10 KB |

1.txt | 1.txt

File Edit View

File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: now i want to make it into an an an.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

This is the Desktop folder:

Directory: 0
Directory: Msc
Directory: New folder
Directory: Pneu
File: l.txt

Ln 70, Col 1 | 100% | Windows (CRLF) | UTF-8

To be able to get and lo the contents the download folder, I had to hard Code the path D:\Downloads

```c
#include <stdio.h>
#include <windows.h>
#include <string.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);

int main()
{
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\0';

    wprintf(L"Listing directories and files in D:\\Documents\n(Current Code File Location: %ls)\n", currentCodeFilePath);

    wchar_t logFileName[FILENAME_MAX];
    _snwprintf(logFileName, FILENAME_MAX, L"%lsbin\\1.txt", currentCodeFilePath);

    // Check if the 'bin' folder exists, if not, create it
    wchar_t binFolderPath[FILENAME_MAX];
    _snwprintf(binFolderPath, FILENAME_MAX, L"%lsbin", currentCodeFilePath);
    if (!CreateDirectoryW(binFolderPath, NULL))
    {
        DWORD error = GetLastError();
        if (error != ERROR_ALREADY_EXISTS)
        {
            wprintf(L"Failed to create 'bin' folder: %ls\n", binFolderPath);
            return 1;
        }
    }

    FILE *logFile = _wfopen(logFileName, L"a"); // Open in "append" mode instead of "write" mode
    if (logFile)
    {
        listDirectoriesAndFilesInRoot(L"D:\\Downloads", logFile);

        fclose(logFile);
        wprintf(L"Successfully logged to %ls.\n", logFileName);

        // Read and print the contents of the log file to the console
        wprintf(L"\nLogged Contents:\n");
        FILE *readLogFile = _wfopen(logFileName, L"r");
        if (readLogFile)
        {
            wchar_t buffer[512];
            while (fgetws(buffer, 512, readLogFile))
            {
                wprintf(L"%ls", buffer);
            }
            fclose(readLogFile);
        }
        else
        {
            wprintf(L"Failed to read %ls.\n", logFileName);
```

```
        }
    }
    else
    {
        wprintf(L"Failed to open or write to %ls.\n", logFileName);
    }

    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        wprintf(L"Error finding directories and files in: %ls\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %ls\n", findFileData.cFileName);
                fwprintf(logFile, L"Directory: %ls\n", findFileData.cFileName); // Log the directory to the file
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
                    wprintf(L"File: %ls\n", findFileData.cFileName);
                    fwprintf(logFile, L"File: %ls\n", findFileData.cFileName); // Log the file to the file
                }
            } while (FindNextFileW(hFind, &findFileData) != 0);

            FindClose(hFind);
        }
```

Untitled | email2.txt | log.txt

File Edit View

File: Subject_ Request for Financial Assistance and Demonstration of Programming Skills (1).pdf
File: Swae Lee LIVE @ Rolling Loud LA 2021.mp4
File: Switch between cameras in blender (Add And Use Multiple Cameras).mp4
File: Switch between cameras in blender (Add And Use Multiple Cameras).vtt
File: Symbol Used in Hydraulic And Pneumatic system (Flow Control Valve,Accumulator,Pipe Lines,Heat Ex).mp4
File: TFT LCD ILI9341 Simulation with arduino.mp4
File: THAT SOUNDS FILTHY! MAKING AN EDM TRAP BEAT IN FL STUDIO!.mp4
File: THAT SOUNDS FILTHY! MAKING AN EDM TRAP BEAT IN FL STUDIO!.vtt
File: The Beat Formula That Helped Kendrick Become KING.mp4
File: The Beauty of Lagrangian Mechanics (SoME2).vtt
File: The Beauty of Lagrangian Mechanics (SoME2)_2.mp4
File: The Denavit-Hartenberg Convention - Robotic Systems.vtt
File: The Denavit-Hartenberg Convention _ Robotic Systems.mp4
File: The equation of a wave - Physics - Khan Academy.mp4
File: The equation of a wave - Physics - Khan Academy.vtt
File: The Equation That Explains (Nearly) Everything!.vtt
File: The fastest spacecraft ever launched.mp4
File: The Geometry of Music.mp4
File: The Latest in AI Music & How to Create Your Own.mp4
File: The Math behind (most) 3D games - Perspective Projection.mp4
File: The Math behind (most) 3D games - Perspective Projection.vtt
File: The Math Behind Music and Sound Synthesis.mp4
File: The Math Behind Music and Sound Synthesis.vtt
File: The Mathematical Problem with Music, and How to Solve It.mp4
File: THE MONEY WILL COME BY ITSELF. The Proven Way to Wealth John D. Rockefeller.mp4
File: The Pythagorean comma.mp4
File: The Pythagorean comma.vtt
File: The Shady Past Of Cardi B.mp4
File: The Shady Past Of Cardi B.vtt

Ln 1, Col 1 | 100% | Windows (CRLF) | ANSI

f