**EX 14:Construct a C program to organise the file using a single level directory.**

**Aim:**

To construct a C program that simulates organizing files using a **single-level directory structure**.

**Algorithm:**

1. **Initialize Data Structures**:
   * Use an array to store file names.
   * Keep track of the number of files.
2. **Display Menu**:
   * Show options for file operations (add, search, delete, display, exit).
3. **Handle User Input**:
   * **Add File**:
     + Check if the file name already exists.
     + Add the file to the array if space is available.
   * **Search File**:
     + Traverse the array to check if the file exists.
   * **Delete File**:
     + Locate the file and remove it from the array.
   * **Display Files**:
     + Print all file names.
4. **Repeat Until Exit**.

**Program:**

#include <stdio.h>

#include <string.h>

#define MAX\_FILES 100

#define MAX\_NAME\_LENGTH 50

// Structure to store files in the directory

typedef struct {

char files[MAX\_FILES][MAX\_NAME\_LENGTH];

int file\_count;

} SingleLevelDirectory;

// Function to add a file

void add\_file(SingleLevelDirectory\* directory, char\* file\_name) {

// Check if the file already exists

for (int i = 0; i < directory->file\_count; i++) {

if (strcmp(directory->files[i], file\_name) == 0) {

printf("File '%s' already exists.\n", file\_name);

return;

}

}

// Add file if there is space

if (directory->file\_count < MAX\_FILES) {

strcpy(directory->files[directory->file\_count], file\_name);

directory->file\_count++;

printf("File '%s' added successfully.\n", file\_name);

} else {

printf("Directory is full. Cannot add more files.\n");

}

}

// Function to search for a file

void search\_file(SingleLevelDirectory\* directory, char\* file\_name) {

for (int i = 0; i < directory->file\_count; i++) {

if (strcmp(directory->files[i], file\_name) == 0) {

printf("File '%s' found at position %d.\n", file\_name, i + 1);

return;

}

}

printf("File '%s' not found.\n", file\_name);

}

// Function to delete a file

void delete\_file(SingleLevelDirectory\* directory, char\* file\_name) {

for (int i = 0; i < directory->file\_count; i++) {

if (strcmp(directory->files[i], file\_name) == 0) {

// Shift remaining files

for (int j = i; j < directory->file\_count - 1; j++) {

strcpy(directory->files[j], directory->files[j + 1]);

}

directory->file\_count--;

printf("File '%s' deleted successfully.\n", file\_name);

return;

}

}

printf("File '%s' not found. Cannot delete.\n", file\_name);

}

// Function to display all files

void display\_files(SingleLevelDirectory\* directory) {

if (directory->file\_count == 0) {

printf("No files in the directory.\n");

return;

}

printf("Files in the directory:\n");

for (int i = 0; i < directory->file\_count; i++) {

printf("%d. %s\n", i + 1, directory->files[i]);

}

}

int main() {

SingleLevelDirectory directory;

directory.file\_count = 0;

int choice;

char file\_name[MAX\_NAME\_LENGTH];

while (1) {

// Display menu

printf("\nSingle-Level Directory\n");

printf("1. Add File\n");

printf("2. Search File\n");

printf("3. Delete File\n");

printf("4. Display Files\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the file name to add: ");

scanf("%s", file\_name);

add\_file(&directory, file\_name);

break;

case 2:

printf("Enter the file name to search: ");

scanf("%s", file\_name);

search\_file(&directory, file\_name);

break;

case 3:

printf("Enter the file name to delete: ");

scanf("%s", file\_name);

delete\_file(&directory, file\_name);

break;

case 4:

display\_files(&directory);

break;

case 5:

printf("Exiting program.\n");

return 0;

default:

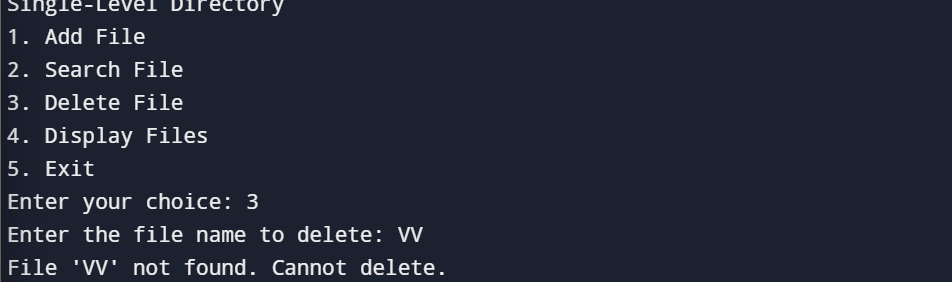
printf("Invalid choice. Please try again.\n");

}

}

}

**OUTPUT:**

****