**K. SWARNA VARSHINI – 192324081**

15.Design a C program to organise the file using a two level directory structure.

**Aim:**

To design a C program that simulates the organization of files using a two-level directory structure.

**Algorithm:**

1. Start the program.
2. Create a structure representing a directory, containing:
   * A directory name.
   * An array of files.
3. Prompt the user to:
   * Create a directory.
   * Add files to a directory.
   * Search for files in a directory.
4. Perform the necessary operations based on user input:
   * Add files to a specific directory.
   * List files in a specific directory.
   * Search for a file in a specific directory.
5. Continue until the user exits.
6. End the program.

**Procedure:**

1. Define a struct for directories and files.
2. Use an array of directories to simulate the two-level structure.
3. Write functions to add directories, add files to directories, display directory contents, and search for files.
4. Use a menu-driven approach to handle user input and call corresponding functions.

### Code:

### #include <stdio.h>

### #include <string.h>

### #define MAX\_DIRS 10

### #define MAX\_FILES 10

### typedef struct {

### char name[20];

### char files[MAX\_FILES][20];

### int file\_count;

### } Directory;

### int main() {

### Directory dirs[MAX\_DIRS];

### int dir\_count = 0, choice;

### char dir\_name[20], file\_name[20];

### int i, j, found;

### do {

### printf("\n1. Create Directory\n2. Add File\n3. Display Directory\n4. Search File\n5. Exit\nEnter choice: ");

### scanf("%d", &choice);

### switch (choice) {

### case 1:

### if (dir\_count < MAX\_DIRS) {

### printf("Enter directory name: ");

### scanf("%s", dirs[dir\_count].name);

### dirs[dir\_count].file\_count = 0;

### dir\_count++;

### } else {

### printf("Maximum directories reached.\n");

### }

### break;

### case 2:

### printf("Enter directory name: ");

### scanf("%s", dir\_name);

### found = 0;

### for (i = 0; i < dir\_count; i++) {

### if (strcmp(dirs[i].name, dir\_name) == 0) {

### if (dirs[i].file\_count < MAX\_FILES) {

### printf("Enter file name: ");

### scanf("%s", dirs[i].files[dirs[i].file\_count]);

### dirs[i].file\_count++;

### } else {

### printf("Maximum files in this directory reached.\n");

### }

### found = 1;

### break;

### }

### }

### if (!found) printf("Directory not found.\n");

### break;

### case 3:

### printf("Enter directory name: ");

### scanf("%s", dir\_name);

### found = 0;

### for (i = 0; i < dir\_count; i++) {

### if (strcmp(dirs[i].name, dir\_name) == 0) {

### printf("Directory: %s\n", dirs[i].name);

### printf("Files:\n");

### for (j = 0; j < dirs[i].file\_count; j++) {

### printf("- %s\n", dirs[i].files[j]);

### }

### found = 1;

### break;

### }

### }

### if (!found) printf("Directory not found.\n");

### break;

### case 4:

### printf("Enter directory name: ");

### scanf("%s", dir\_name);

### printf("Enter file name: ");

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

### found = 0;

### for (i = 0; i < dir\_count; i++) {

### if (strcmp(dirs[i].name, dir\_name) == 0) {

### for (j = 0; j < dirs[i].file\_count; j++) {

### if (strcmp(dirs[i].files[j], file\_name) == 0) {

### printf("File found in directory %s.\n", dir\_name);

### found = 1;

### break;

### }

### }

### }

### if (found) break;

### }

### if (!found) printf("File not found.\n");

### break;

### case 5:

### printf("Exiting...\n");

### break;

### default:

### printf("Invalid choice.\n");

### }

### } while (choice != 5);

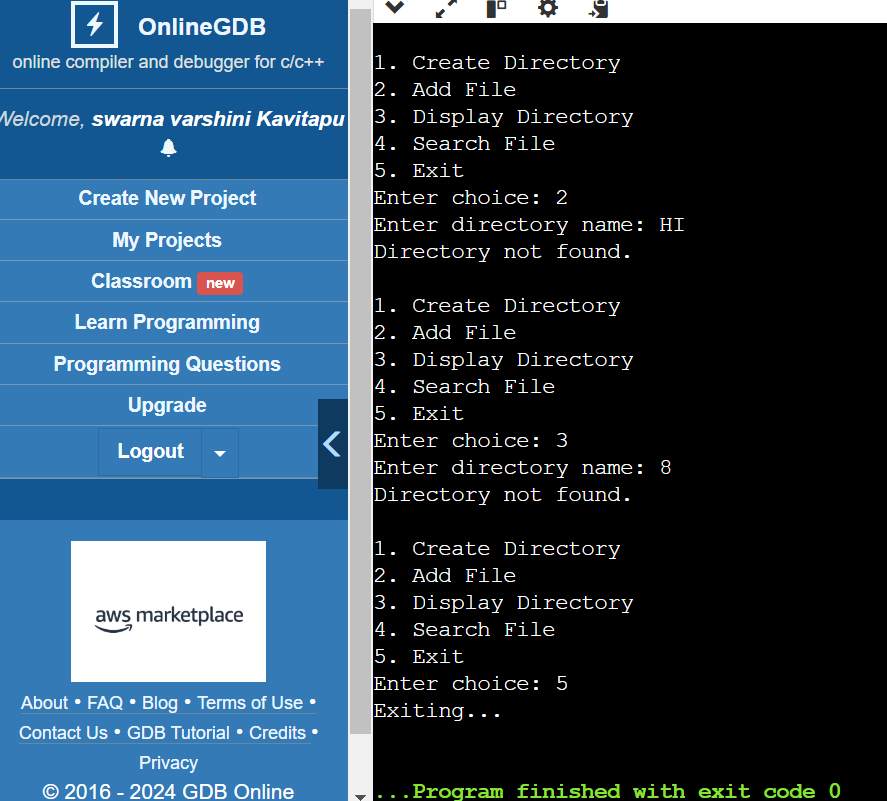
### return 0;

### }

**Result:**

1. Successfully created directories.
2. Added files to directories.
3. Displayed the contents of a directory.
4. Searched and found files in a specific directory.

**Output:**

****