Lab 1 - Implement Matrix manipulation

Consider the 2D representation for your chosen domain. Perform all data structure operations (insertion, Deletion, linear search) using 2D arrays for any chosen logical data of your domain. Implement any two matrix operations.

#include <stdio.h>

#include <stdlib.h>

#define ROWS 3

#define COLS 3

// #define COUNT 2

struct matrixStruct

{

char hotel\_name[25];

int bookings[ROWS][COLS];

};

struct matrixStruct hotel;

void insertionDisplay();

int deleteElement(int row, int col);

void displayMatrix();

int linearSearch(int value);

void addMatrix();

void mulMatrix();

int main()

{

int choice, value, count = ROWS \* COLS, row, col;

int continueMenu = 1; // A flag to continue the menu loop

while (continueMenu)

{

printf("----------------------Hotel\_Management---------------------------\n");

printf("Enter your Choice (1: Insert, 2: Delete 3: Display, 4: Searching,5 : Add Matrix, 6 : multiply matrix, 0: Exit): ");

scanf("%d", &choice);

switch (choice)

{

case 1:

// Insert the hotel name

printf("Enter Hotel Name :");

scanf("%s", &hotel.hotel\_name);

printf("Enter the booking count received in the last %d days:\n", count);

insertionDisplay();

break;

case 2:

// Delete the matrix

printf("Enter the Row and Column of the matrix : ");

scanf("%d %d", &row, &col);

deleteElement(row - 1, col - 1);

break;

case 3:

displayMatrix();

break;

case 4:

printf("Enter the value to be searched :");

scanf("%d", &value);

linearSearch(value);

break;

case 5:

printf("----------------MATRIX ADDITION----------------------");

addMatrix();

break;

case 6:

printf("----------------MATRIX MUL----------------------");

mulMatrix();

break;

case 0:

continueMenu = 0; // Exit the loop

break;

default:

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

}

}

return 0;

}

void addMatrix()

{

// adding two matrices

int i, j, sum[3][3];

for (i = 0; i < ROWS; ++i)

for (j = 0; j < COLS; ++j)

{

sum[i][j] = hotel.bookings[i][j] + hotel.bookings[i][j];

}

// DIsplay the matrix

for (int i = 0; i < ROWS; i++)

{

printf(" ");

for (int j = 0; j < COLS; j++)

{

printf("%d ", sum[i][j]);

}

printf("\n");

}

}

void mulMatrix()

{

// adding two matrices

int i, j, mul[3][3];

for (i = 0; i < ROWS; ++i)

for (j = 0; j < COLS; ++j)

{

mul[i][j] = hotel.bookings[i][j] \* hotel.bookings[i][j];

}

// DIsplay the matrix

for (int i = 0; i < ROWS; i++)

{

printf(" ");

for (int j = 0; j < COLS; j++)

{

printf("%d ", mul[i][j]);

}

printf("\n");

}

}

void displayMatrix()

{

printf("----------------------3x3 Matrix of bookings---------------------------\n");

for (int i = 0; i < ROWS; i++)

{

printf(" ");

for (int j = 0; j < COLS; j++)

{

printf("%d ", hotel.bookings[i][j]);

}

printf("\n");

}

}

void insertionDisplay()

{

// Insertion hotel booking details

for (int i = 0; i < ROWS; i++)

{

for (int j = 0; j < COLS; j++)

{

scanf("%d", &hotel.bookings[i][j]);

}

}

// Display

printf("----------------------Bookings of %s---------------------------\n", hotel.hotel\_name);

for (int i = 0; i < ROWS; i++)

{

printf(" ");

for (int j = 0; j < COLS; j++)

{

printf("%d ", hotel.bookings[i][j]);

}

printf("\n");

}

}

// Delete

int deleteElement(int row, int col)

{

if (row >= 0 && row < ROWS && col >= 0 && col < COLS)

{

hotel.bookings[row][col] = -1; // Mark the element as deleted

return printf("Element deleted\n");

}

return printf("Element not found\n");

}

// Linear search

int linearSearch(int value)

{

printf("----------------------------Searching----------------------------------\n");

for (int i = 0; i < ROWS; i++)

{

for (int j = 0; j < COLS; j++)

{

if (hotel.bookings[i][j] == value)

{

return printf("value %d is found\n", value);

}

}

}

return printf("value is not found\n");

}

OUTPUT :

