//travel planner

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Destination structure

struct Destination {

char name[100];

char description[500];

};

// Itinerary structure

struct Itinerary {

char destination[100];

char date[20];

struct Itinerary\* next;

};

// Function to add a destination

void addDestination(struct Destination destinations[], int\* destCount) {

if (\*destCount < 10) {

struct Destination dest;

printf("Enter destination name: ");

scanf("%s", dest.name);

printf("Enter destination description: ");

scanf(" %[^\n]", dest.description);

destinations[\*destCount] = dest;

(\*destCount)++;

printf("Destination added successfully!\n");

} else {

printf("You've reached the maximum number of destinations.\n");

}

}

// Function to view destinations

void viewDestinations(struct Destination destinations[], int destCount) {

if (destCount == 0) {

printf("No destinations available. Please add destinations first.\n");

return;

}

printf("\nDestinations:\n");

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

printf("| Name | Description |\n");

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

for (int i = 0; i < destCount; i++) {

printf("| %-20s | %-46s |\n", destinations[i].name, destinations[i].description);

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

}

printf("\n");

}

// Function to create an itinerary (linked list)

void createItinerary(struct Itinerary\*\* head, struct Destination destinations[], int destCount) {

if (destCount == 0) {

printf("No destinations available. Please add destinations first.\n");

return;

}

if (\*head == NULL) {

struct Itinerary\* newItinerary = (struct Itinerary\*)malloc(sizeof(struct Itinerary));

if (newItinerary == NULL) {

printf("Memory allocation failed.\n");

return;

}

printf("Enter destination name for the itinerary: ");

scanf("%s", newItinerary->destination);

printf("Enter date for the itinerary: ");

scanf("%s", newItinerary->date);

newItinerary->next = NULL;

\*head = newItinerary;

printf("Itinerary created successfully!\n");

} else {

printf("You've reached the maximum number of itineraries.\n");

}

}

// Function to view itineraries (linked list)

void viewItineraries(struct Itinerary\* head) {

if (head == NULL) {

printf("No itineraries available.\n");

return;

}

printf("\nItineraries:\n");

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

printf("| Destination | Date |\n");

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

struct Itinerary\* current = head;

while (current != NULL) {

printf("| %-20s | %-20s |\n", current->destination, current->date);

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

current = current->next;

}

printf("\n");

}

// Function to free memory used by the linked list

void freeItineraries(struct Itinerary\* head) {

struct Itinerary\* current = head;

while (current != NULL) {

struct Itinerary\* temp = current;

current = current->next;

free(temp);

}

}

// Function to sort destinations alphabetically

void sortDestinations(struct Destination destinations[], int destCount) {

if (destCount > 1) {

int i, j;

struct Destination temp;

for (i = 0; i < destCount - 1; i++) {

for (j = i + 1; j < destCount; j++) {

if (strcmp(destinations[i].name, destinations[j].name) > 0) {

temp = destinations[i];

destinations[i] = destinations[j];

destinations[j] = temp;

}

}

}

printf("Destinations sorted alphabetically.\n");

} else {

printf("There are not enough destinations to sort.\n");

}

}

int main() {

struct Destination destinations[10];

struct Itinerary\* itineraries = NULL;

int destCount = 0;

int choice;

while (1) {

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\* Travel Planner Menu \*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

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

printf("2. View Destinations\n");

printf("3. Create Itinerary\n");

printf("4. View Itineraries\n");

printf("5. Sort Destinations\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addDestination(destinations, &destCount);

break;

case 2:

viewDestinations(destinations, destCount);

break;

case 3:

createItinerary(&itineraries, destinations, destCount);

break;

case 4:

viewItineraries(itineraries);

break;

case 5:

sortDestinations(destinations, destCount);

break;

case 6:

freeItineraries(itineraries);

return 0;

default:

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

}

}

return 0;

}