Nikeem Dunkelly-Allen, ECE322, Assignment 1

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Structure to store data

struct NamedPlace {

int numcode;

char sab[3];

char name[100];

int pop;

float area;

float lati;

float longi;

int roadcode;

float dist;

struct NamedPlace\* next;

};

// Function to create a new node

struct NamedPlace\* createNode(int numcode, char\* sab, char\* name, int pop, float area, float lati, float longi, int roadcode, float dist)

{

struct NamedPlace\* newNode = (struct NamedPlace\*)malloc(sizeof(struct NamedPlace));

newNode->numcode = numcode;

strcpy(newNode->sab, sab);

strcpy(newNode->name, name);

newNode->pop = pop;

newNode->area = area;

newNode->lati = lati;

newNode->longi = longi;

newNode->roadcode = roadcode;

newNode->dist = dist;

newNode->next = NULL;

return newNode;

}

// Function to add a new node to the list

struct NamedPlace\* addNode(struct NamedPlace\* head, int numcode, char\* sab, char\* name, int pop, float area, float lati, float longi, int roadcode, float dist)

{

if(head == NULL)

return createNode(numcode, sab, name, pop, area, lati, longi, roadcode, dist);

struct NamedPlace\* temp = head;

while(temp->next != NULL)

temp = temp->next;

temp->next = createNode(numcode, sab, name, pop, area, lati, longi, roadcode, dist);

return head;

}

// Function to search for a given place name in the list

struct NamedPlace\* search(struct NamedPlace\* head, char\* placeName, char\* sab)

{

//struct NamedPlace\* temp = head;

while(head != NULL)

{

if(strcmp(head->name, placeName) == 0 && strcmp(head->sab, sab) ==0)

return head;

head = head->next;

}

return NULL;

}

// Function to print information of a given node

void printNode(struct NamedPlace\* node)

{

printf("Numeric Code: %d\n", node->numcode);

printf("State Abbreviation: %s\n", node->sab);

printf("Name: %s\n", node->name);

printf("pop: %d\n", node->pop);

printf("Area: %.2f\n", node->area);

printf("lati: %.2f\n", node->lati);

printf("longi: %.2f\n", node->longi);

printf("Road Code: %d\n", node->roadcode);

printf("Distance to Intersection: %.2f\n", node->dist);

printf("\n");

}

int main()

{

printf("File is being read... \n");

// Initialize list

struct NamedPlace\* head = NULL;

// Read data from file

FILE\* fp = fopen("/home/www/class/een318/named-places.txt", "r");

if(fp == NULL)

{

printf("Error: File not found!");

exit(0);

}

char line[200];

while(fgets(line, sizeof(line), fp))

{

int numcode;

char sab[3];

char name[100];

int pop;

float area;

float lati;

float longi;

int roadcode;

float dist;

sscanf(line, "%8d %2s %s %d %f %f %f %d %f", &numcode, sab, name, &pop, &area, &lati, &longi, &roadcode, &dist);

head = addNode(head, numcode, sab, name, pop, area, lati, longi, roadcode, dist);

// trim trailing spaces from name

int i = strlen(name) - 1;

while(name[i] == ' ')

{ name[i] = '\0';

i--;

}

}

fclose(fp);

printf("File read succesful \n");

// Enter interactive loop

char placeName[100];

char sab[3];

while(1)

{

printf("Enter city/town name: ");

scanf("%s", placeName);

struct NamedPlace\* temp = head;

printf("The place name appears in the following states: \n");

while(temp != NULL)

{

if(strcmp(temp->name, placeName) == 0)

printf("%s ", temp->sab);

temp = temp->next;

}

printf("\n");

printf("Enter a state abbreviation (ex: California = CA): ");

scanf("%s", sab);

temp = search(head,placeName, sab);

printf("Information for choosen place: \n");

printNode(temp);

printf("\n");

printf("Enter 0 to exit or 1 to continue: ");

int flag;

scanf("%d", &flag);

if(flag == 0)

break;

}

return 0;

}