#### <TITLE OF THE PROJECT>

A PROJECT REPORT

***Submitted by***

#### <BL.EN.UNEAC19071> < Suryajith Sujith>

#### As a part of

#### <19EAC203> <Data structures and algorithms>

Offered by

**< Mr. Nandi Vardhan Harlalli Rajendra>**

Department of Electronics and Communications Engineering



AMRITA SCHOOL OF ENGINEERING, BENGALURU CAMPUS

AMRITA VISHWA VIDYAPEETHAM

#### BENGALURU 560 035

OCTOBER-2020

**CONTENTS:**

1. Problem Definition

2. Specifications and Flow

3. Design

4. Implementation

5. Results

6. Conclusion

7. References

**Problem statement:**

Implement a city database. Each database record contains the name of the city (a string of arbitrary length) and the coordinates of the city expressed as integer x and y coordinates. Your database should allow records to be inserted, deleted by name or coordinate, and searched by name or coordinate. Another operation that should be supported is to print all records within a given distance of a specified point (capital city).

Ex:

City should be having following description and should be searched either by name or using x and y (both). The distance from the capital city has to be captured.

Name: Bengaluru

Coordinate→X→22 (Latitude)

Coordinate→Y→76 (Longitude)

Distance from capital: 0 km

A student can create a city structure with required attribute and use any data structure of choice.

**Specifications and flow:**

Operations required

1. Insertion of an element in singly linked list:

This operation is used for inserting the data i.e name of city and its latitude and longitude into the database.

2. Deletion of an element:

This operation is used to delete a particular data from the database.

3. Displaying/printing elements of the list:

This operation is used to display the contents of the database.

4. Searching elements in the list:

This operation is used for the purpose of searching a particular data in the database created.

5. Calculating distance which requires a mathematical formula from 2-D coordinate geometry:

In this operation the mathematical formulae from 2-D coordinate geometry is used.

NOTE:

Haversine formula is a method for calculating distance between 2 cities with much better accuracy.

**Design:**

Functions used: -

void insert ()

//Function used to insert a node

void new\_distance()

//Used to find the distance between two cities with it's coordinates and to enable overwritting

void delete()

//Used to delete node by name or by coordinates

void print()

//Used for printing the node data

void search()

//Function used for searching the node by name or it's coordinates

void Range()

//Used to print cities within a range

Choice of data structure: Singly linked lists

Reason to use singly linked lists:

As a matter of fact, I personally feel a lot easier to work with singly linked lists.

**Implementation:**

#include <stdio.h>      //Header files

#include <stdlib.h>

#include <math.h>

#include <string.h>

// Singly linked list node

typedef struct city{

char name[15];           // City\_Name

float x;                 // x - Latitude

float y;                 // y - Longitude

float dist;              // Distance from capital

struct city\*next;        // Node pointer

}City;

City \*root,\*head=NULL,\*temp=NULL;

//List of functions used

void insert();   //insertion operation

void delete();   //deletion operation

void print();    //to display the inserted data

void new\_distance(); //for overwritting

void search();   //seraching operation

void Range();    //to print cities within a range

void main(){

int q, ch;

while(1){

printf(" 1.Insert\n 2.Delete\n 3.Print \n 4.Search\n 5.Find cities available within a range\n");

printf("Enter your choice: ");

scanf("%d",&q);

switch(q){

case 1:

insert();

break;

case 2:

delete();

new\_distance();   //to enable Overwritting if capital is deleted

break;

case 3:

print();

break;

case 4:

search();

break;

case 5:

Range();

break;

default:

printf("\nEnter your choice: ");

}

printf("\nEnter \n1.To continue \n0.To exit \n");

scanf("%d",&ch);

if(ch==1)

continue;

else

break;

}

}

void insert()         //Function used to insert a node

{

root = (City\*) malloc( sizeof(City) );    //Dynamic Memory allocation

printf("\nEnter the name of the city: ");

scanf("%s",root->name);              //User input for city name

printf("\nEnter latitude: ");

scanf("%f",&root->x);                //user input for latitude

printf("\nEnter Longitude: ");

scanf("%f",&root->y);                //user input for longitude

root->next=NULL;

if(head==NULL){

head=root;

temp=root;

//formulae to calculate distance = sqrt( (x2-x1)\*(x2-x1) + (y2-y1)\*(y2-y1) )

temp->dist=sqrt((temp->x-head->x)\*(temp->x-head->x) + (temp->y-head->y)\*(temp->y-head->y));

}

else{

temp->next=root;

temp=root;

temp->next=NULL;

temp->dist=sqrt((temp->x-head->x)\*(temp->x-head->x) + (temp->y-head->y)\*(temp->y-head->y));

}

printf("\n");

}

void new\_distance()    //Used to find the distance between two cities with it's coordinates

{

temp=head;

while(temp!=NULL){

 //formulae to calculate distance = sqrt ( (x2-x1)\*(x2-x1) + (y2-y1)\*(y2-y1) )

temp->dist=sqrt((temp->x-head->x)\*(temp->x-head->x) + (temp->y-head->y)\*(temp->y-head->y));

temp =temp->next;

}

temp=head;

}

void delete()         //Used to delete node by name or by coordinates

{

temp=head;

char d[50];

float latitude ,longitude;

int k;

City \*ptr;

ptr=head;

printf("\nEnter \n 1.To Delete by entering city name  \n 0.To Delete by entering latitude and longitude\n");

scanf("%d",&k);

if(k==1){

printf("\nEnter the Name of the city: ");

scanf("%s",d);

while(temp!=NULL&&strcmp(temp->name,d)!=0){

ptr=temp;

temp=temp->next;

}

// If head node holds the name to be deleted

if(temp==head){

head=temp->next;// Modified head

temp->next=NULL;

free(temp);     // Free memory

}

//If name is at last node

else if(temp->next==NULL){

ptr->next=NULL;

free(temp);

}

else{

ptr->next=temp->next;

temp->next=NULL;

free(temp);

}

}

else if(k==0){

printf("\nEnter the Latitude and Longitude of the city\n");

printf("Latitude: ");

printf("\n");

scanf("%f",&latitude);

printf("Longitude: ");

printf("\n");

scanf("%f",&longitude);

while((temp->x!=latitude)&&(temp->y!=longitude)){

ptr=temp;

temp=temp->next;

}

if(temp==head){

head=temp->next;

temp->next=NULL;

free(temp);    //free()- deallocates the memory previously allocated

}

else{

ptr->next=temp->next;

temp->next=NULL;

free(temp);

}

}

}

void print() //Used for printing the node data

{

temp=head;

City \*p;

while(temp!=NULL){

p = temp;

printf("\n%s\t",temp->name);

printf("%f° N\t%f° E\t%f km",temp->x,temp->y,temp->dist);

printf("\n");

temp = temp->next;

}

temp = p;

}

void search()     //Function used for searching the node by name or it's coordinates

{

int s;

temp=head;

printf("\nEnter \n1.To search by name \n0.To search by coordinates\n");

scanf("%d",&s);

if(s==1){

char input\_city\_name[50];

printf("\nEnter city name: ");

scanf("%s",input\_city\_name);

while(temp!=0 && strcmp(temp->name,input\_city\_name)!=0){

temp=temp->next;

}

if(temp!=0){

printf("\nCity exists\n");

printf("\n%s\t%f° N\t%f° E\t%f km",temp->name,temp->x,temp->y,temp->dist);

}

else{

printf("\nCity not found in database\n");

}

}

else if(s==0){

float lt,ln;

printf("\nEnter Latitude: ");

scanf("%f",&lt);

printf("\nEnter longitude: ");

scanf("%f",&ln);

while(temp!=0 && temp->x!=lt && temp->y!=ln){

temp=temp->next;

}

if(temp->x==lt && temp->y==ln){

printf("\nCoordinates found\n");

printf("\n%s\t%f° N\t%f° E\t%f km",temp->name,temp->x,temp->y,temp->dist);

}

else{

printf("\nCoordinates aren't there on databse\n");

}

}

temp=head;

printf("\n");

}

void Range()     //Used to print cities within a range

{

int range;

temp = head;

printf("\nEnter range: ");

scanf("%d",&range);

while(temp!=NULL){

if(temp->dist<=range){

printf("\n%s \t%f° N\t%f° E\t%f km",temp->name,temp->x,temp->y,temp->dist);

}

temp=temp->next;

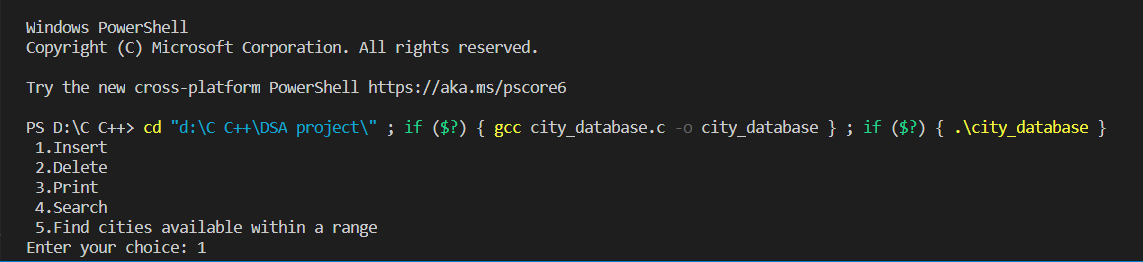
}

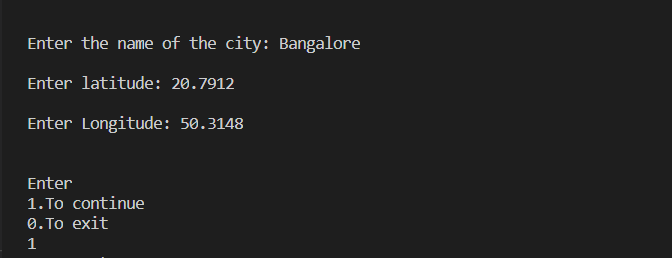
temp=head;

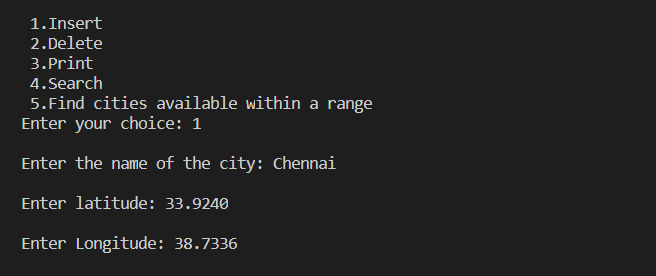
printf("\n");

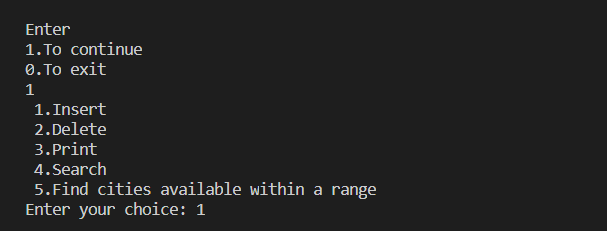
}

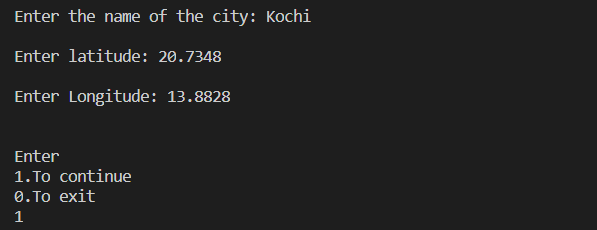
**Output:**

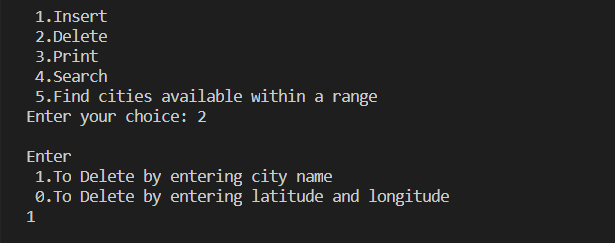


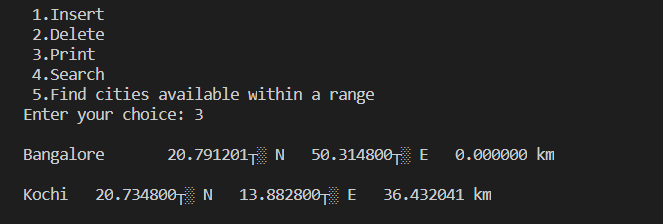


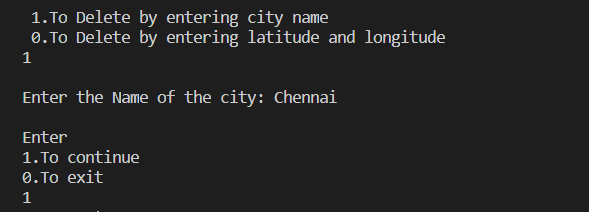


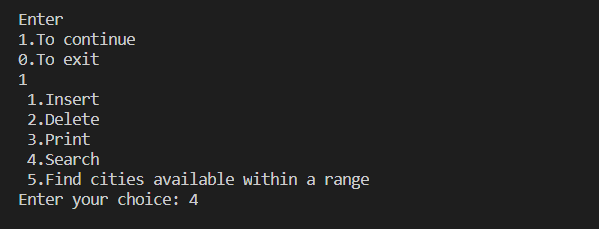
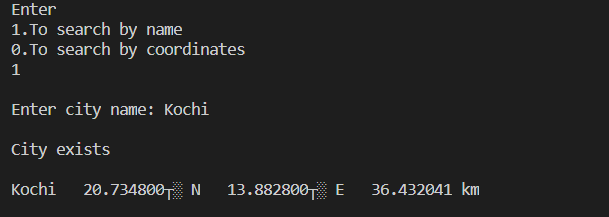
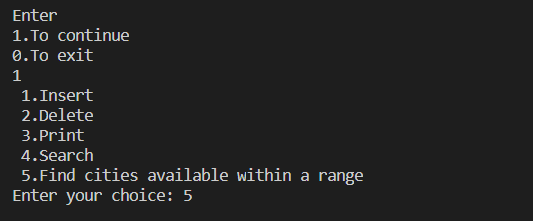


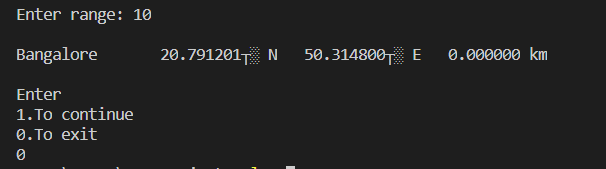












As we see the output for the distance between cities is displayed incorrectly, this would be because of the fact that degrees weren’t converted to radians.

**Conclusion**

Complexity for all the functions implemented is O(n).

References:

<https://www.learncbse.in/free-ncert-solutions-class-11th-maths-chapter-10-straight-lines/>

<https://www.learn-c.org/en/Linked_lists#:~:text=What%20is%20a%20linked%20list,next%20member%20of%20the%20list>

<https://www.geeksforgeeks.org/swap-nodes-in-a-linked-list-without-swapping-data/>

<https://www.geeksforgeeks.org/menu-driven-program-using-switch-case-c/>

<https://www.studytonight.com/c/programs/misc/menu-driven-program>