ENGG1410 – December 02, 2022

Lab Mini Project #2 – GPS

Group #34 – Aiman Kassam

Problem Statement

The objective(s) of this program are as follows:

- To read values from a text file using the *fscanf()* and use the obtained values to perform mathematical operations
- To calculate the distance between any chosen number of people based on any given position coordinates
- Using calculated distances to determine which of those users are closest to yourself
- Becoming accustomed to using structs and custom-built functions
- To learn through practice a basic foundation of the programming skills that go into building a GPS

The goal of this program is to take values for position and time coordinates with associated names either directly through user input, or via reading a text file. The program would use these values to figure out how far away someone is, as well as which of the people entered would be the closest to yourself. Operations for calculating distance and taking user input would be used *many* times, forcing us *(the programmer(s))* to create custom functions with appropriate parameters. Using custom functions in accompaniment with for loops allows for highly efficient calculations for all distances. Combining these with struct arrays allows for easy adaptability regardless of how many users you wish to compare, whether its 3, 20, 600,000, or anything in between.

We are provided with exactly which functions we are expected to create, and in which order we are expected to call them. However, it is on us to determine the most efficient, concise, accurate, and fool-proof way to code the functions.

Assumptions & Constraints

Assumptions made & Constraints enforced when coding this program:

- User will input an integer number when asked for number of users (no characters of any kind, no fractions/decimals, etc.)
- User will input a valid file directory to a text file.
- Text file is presumably laid out exactly as predicted by the program in the correct order every time.
- Text file must have each value of significance on its own line, no commas or any other form of punctuation.
- Distances assume that all users are relatively close to each other (say within the same city/country)
- Distances assume that all users are located in the south-east

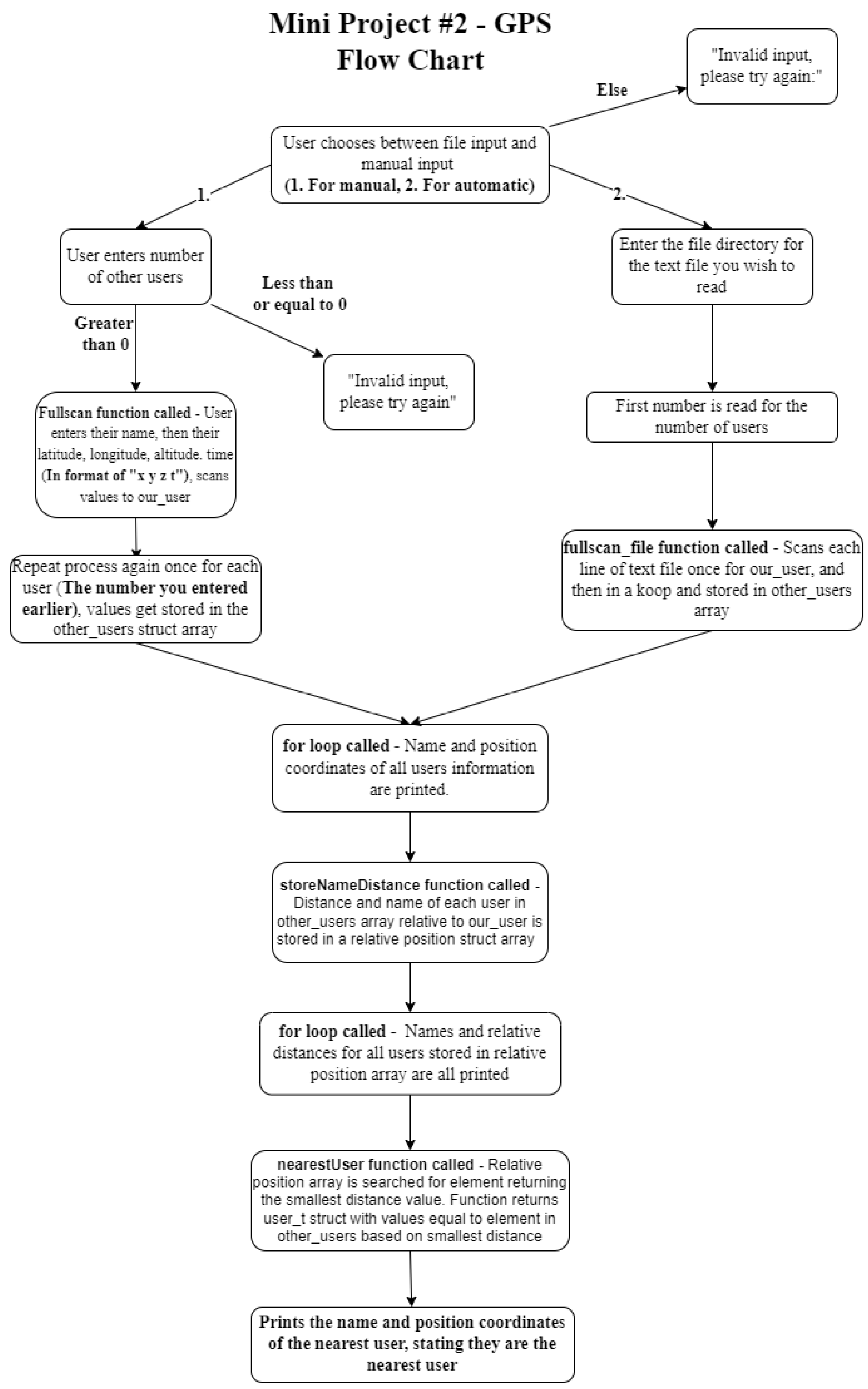
Problem Solving Process

There were two techniques employed to solve the problems presented by this mini project: pseudocode and a flow chart.

Pseudocode:

```
#Import libraries
Declare struct for user_t{
    Stores name of user in string;
    Stores latitude, longitude, altitude, and time in doubles;
}
Declare struct for relative position{
    Stores name of other_user in string;
    Stores distance of other_users each relative to our_users;
}
Declare scan_user function for one person(input user_t struct){
    Scans for name of user;
    Scans for time, latitude, longitude, and altitude coordinates of user;
}
Declare fullscan function(input user_t struct,  user_t array, int number of users){
    Calls scan_user once for one user_t;
    For(I = 0, in range of 0 to less than number of users, i++){
        Scan_user(user_t array[i]);
}
}
Declare scan_user_file(input user_t , file pointer){
    Same function as scan_user, but takes input from file;
}
Declare fullscan_file(input user_t struct, user_t struct array, int number of users, file pointer){
    Same function as fullscan, but takes input from file;
}
Declare distance function(input user_t struct, user_t struct){
    Calculates distance between two user_t structs using sqrt((x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2;
}
Declare storeNameDistance function(input user_t, user_t array, relpos array, int num){
    Calls distance function in for loop for user_t struct and user_t array [i];
    Stores names and distances in relpos array for all [num] users;
}
Declare minElement function (input relative position array, int num){
    Declare int element;
    For(i=0; i < num; i++){
        Search through array until minimum value is found;
        Set element equal to i;
    }
}
Declare function nearestUser(input user_t array, relative position array,  int num, file pointer){
    Declare user_t struct closestUser;
    Calls minElement to find smallest element value from relative position array;
    Sets closestUser equal to other_users[minElement(relpos array)];
    Returns closestUser;
}
Declare void main(){
    Declare int numberofusers, operator;
    Declare struct user_t our_users
    Ask user if they want to input via a file or manually (1. For manual, 2. For automatic), store
number in operator;
    While(op is not any of the provided options){
        Tell the user to input their optionagain
    }
    If(operator is 1){
        Ask for number of users;
        Declare struct user_t our_user, other_users[numberofusers], closestUser;
        Declare struct relativePosition storage[numberofusers];
        Call fullscan(our_user, other_users, numberofusers)
        Calls storeNameDistance function(our_user, other_users, storage, numberofusers)
        Set closestUser = nearestUser(other_users, storage, nunberofusers);
        Print name and position coordinates of the nearest user;
    }
    If(operator is 2){
        Ask for file directory
        Store first number in numberofusers;
        Declare struct user_t our_user, other_users[numberofusers], closestUser;
        Declare struct relativePosition storage[numberofusers];
        Call fullscan_file(our_user, other_users, numberofusers)
        Calls storeNameDistance function(our_user, other_users, storage, numberofusers)
        Set closestUser = nearestUser(other_users, storage, nunberofusers);
        Print name and position coordinates of the nearest user;
    }
}
```

Flow Chart:


Mini Project #2 - GPS Flow Chart
"Invalid input, please try again:"
Else
User chooses between file input and manual input (1. For manual, 2. For automatic)
1.
2.
User enters number of other users
Enter the file directory for the text file you wish to read
Less than or equal to 0
Greater than 0
"Invalid input, please try again"
Fullscan function called - User enters their name, then their latitude, longitude, altitude. time (In format of "x y z t"), scans values to our_user
First number is read for the number of users
Repeat process again once for each user (The number you entered earlier), values get stored in the other_users struct array
fullscan_file function called - Scans each line of text file once for our_user, and then in a koop and stored in other_users array
for loop called - Name and position coordinates of all users information are printed.
storeNameDistance function called - Distance and name of each user in other_users array relative to our_user is stored in a relative position struct array
for loop called - Names and relative distances for all users stored in relative position array are all printed
nearestUser function called - Relative position array is searched for element returning the smallest distance value. Function returns user_t struct with values equal to element in other_users based on smallest distance
Prints the name and position coordinates of the nearest user, stating they are the nearest user

System Overview & Justification of Design

The program makes the user choose between user input and file input before executing anything else. Once chosen, the program takes inputs for its calculations accordingly. The point of this program is to take a list of given users and their position coordinates in comparison to your own and tell you who is closest to you. The basic ideas that go into this program can be the building blocks of a GPS.

What makes this program unique is how well it scales up to any number of users. Certain programs can only be designed for a specific number of users. This program also has a small degree of fool-proofing, in that it will not allow you to enter invalid inputs at certain steps. Those steps include the number of users as well as the operator number which is used to decide whether to use file or manual input.

There are two major custom structures in the program: user_t and relpos (short for relative position). The user_t struct stores a single string, used for a given user's name, and 4 doubles, which store latitude, longitude, altitude, and time. The user_t struct is used twice as a single struct and once as an array. The two individual structs are our_user, representing the user we wish to measure relative distances to, and the closest user, which is given the values of the user with the smallest distance from our_user. The user_t array stores every single other user, along with their 4 double values for position coordinates and time. The fullscan function and fullscan_file functions respectively give these struct variables meaningful values. The storeNameDistance function then takes our_user, all other_users, calculates the distance of each other_user relative to our_user, and stores them all in the single relative position array (named 'storage'). Next, the nearestUser function is called which searches the relative position array, simply looks for the smallest value, then assigns the element representing that smallest value to a variable (called 'element'). It then assigns the values of other_users[element] to the closestUser user_t struct. Finally, two loops are used to list the position coordinates and distance of each user, and then a standalone print statement tells the name and position coordinates of the nearest user.

The scan_user function is called by the fullscan function, which on its own only asks for the position coordinates and name of a single user (same as scan_user_file with fullscan_file). The distance function is called in a loop by the storeNameDistance function, and simply takes to user_t structs and computes sqrt((x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2) with x, y, and z representing latitude, longitude, and altitude. The minElement function is called by the nearestUser function, which returns an int representing the element of an array holding the smallest value.

If 1. Is selected:

1. Store number of users in int variable
2. Call fullscan function

3. Prints all position coordinates of all users in for loop
4. Calls storeNameDistance function, stores names and relative distances in relpos array
5. Calls nearestUser function, stores return value in user_t struct
6. Prints nearest user's name and position coordinates

If 2. Is selected:

1. Store number of users in int variable from first line of text document
2. Call fullscan_file function
3. Prints all position coordinates of all users in for loop
4. Calls storeNameDistance function, stores names and relative distances in relpos array
5. Calls nearestUser function, stores return value in user_t struct
6. Prints nearest user's name and position coordinates

One alternate approach I had thought of for storing position coordinates was to use typedef to define a struct called a Point, which would store three double values representing latitude, longitude, and altitude. For obtaining values from the text file line by line, I had considered using fscanf in the main function, storing the values in variables, and then inputting them directly into the functions performing the calculations. The reason I decided against this was it would dramatically increase the number of variables that would need to be declared and make things harder to keep track of. I also initially intended to make this program only work for the 4 users in the original text file, which would have made the program unable to scale to any other number of users.

Error Analysis

No syntax errors are present in my current program, as it compiles perfectly fine.

No logical errors are present, as all functions perform exactly as they are expected to and scale very nicely.

As it stands now, this program functions perfectly as intended. However, one error did come up under very specific circumstances. For some reason, the minElement function had the problem of returning the incorrect value if the closest user in the relative position array was represented by element 0. Using print statements, I was able to figure out that every time the 0^{th} element was to be returned, minElement returned exactly the number of users. This was a strange effect considering the loop in which the element variable was assigned its value shouldn't even be able to reach the number of users and should stop at 'number of users minus 1'. I was able to implement a fix that would allow the function to perform as intended, which was to implement an if statement that tells the function to set the value of the element variable back to 0 if it ever reaches or exceeds the number of users. That being said, it was merely a band-aid fix, and I would have preferred to find a more elegant solution.