Matthew Hibbin's (mjh32) CS23820 Assignment 2020
Design Document

Design Overview

The program contains a successful implementation of four of the five features outlined in the assignment specification. The features included are:
Feature 1 - Reading GPS data from a file to a linked list
Feature 2 - Generating movement data from the linked list entries
Feature 3 - Outputting the GPS & movement data as a CSV file
Feature 4 - Using configuration files & a menu system to control the program's features
No attempt was made to implement graph generation using the created CSV files.

The "main.c" file implements Feature 4. Program arguments can be given to the program which contain the file name (or path) of the configuration file, allowing the program to run in batch mode. The batch mode will automatically read in the configuration from the file, generating the CSV file using the data from the specified data file. If no program arguments are given, then the program will present the user with a menu of five options: load configuration from a file, load the data & generate CSV files, change the current configuration, save the current configuration to a file or quit. The configuration is stored using global variables for functions to enable easy access to their values. These variables have default values so that the user only needs to provide a data file in order to use the program. They can be modified with the third option in the menu system and saved to a file using the fourth menu option. The program quits after the successful creation of a CSV file.

The "listFunctions.c" file implements Features 1, 2 and 3. Global variables are used in this file to store the top nodes of linked lists. An array is used to store the head nodes, as the program is able to handle multiple sets of data at a time. When passing program arguments, multiple configuration files can be passed in linking to different data files. The program will recognise how many files are being given and create the appropriate amount of linked lists to handle each data set. For each data set the data will be loaded into a linked list, be used in calculations and print the results to a CSV file. The decision to allow multiple configuration files to be processed concurrently enables efficiency, preventing the need for multiple calls to the program. When reading in the data file, each line containing valid GPS data is extracted and put into a linked list node. The node is not yet connected to the linked list, however, it is first compared to the last node created. If the GPS coordinates match, the last node is overwritten with the current node. When the coordinates do not match, the last node is placed at the tail of the linked list and the current node then becomes the 'last node'. The node created from the last line of data in the data file, will always be added to the end of a linked list. The calculation for movement data such as distance, time, speed and behaviour are not stored in the linked list. Rather,

this data is calculated when producing a CSV file to lower the amount of disk space used by the program while it is running.

The “nodeFunctions.c” and “header.c” files implement the linked list structure. The functions used to organise the linked list were adapted from the implementation given in lecture 13. Within the linked list, the Latitude and Longitude are stored as a double type and the date & time as a time_t type. As stated previously, this is the only data stored within the linked list, preventing the use of unnecessary disk space. This was an important consideration, as the data files provided were large for text files. The time_t type was used over the struct tm type as it was more compact, further reducing the disk space that would be needed. To read in or print the time_t data, it must first be converted to the struct tm type.

Known Issues

The program has a few minor design flaws and bugs. Global variables are considered bad practise due to their wide availability. However, returning multiple variables of different types from a function is not possible, making the use of global variables necessary. An alternative solution was attempted, which involved creating a temporary file to hold the values when they were not being used by the current function. This worked, however it was impractical as the code to read in the data had to be duplicated for each function that needed it. A function could not be created to read in the data due to the returning limitations.

Another known issue was limiting the range of the latitude and longitude when it was read in from a file. The specific code for this functionality is commented out and there are no logical flaws within it. However, when it is active, the program crashes when printing any data to the file or terminal. Multiple attempts were made to rectify the problem, such as separating the code to a function, using pointers to the variables used and changing when the code was executed. Even after research the issue remained unresolved and therefore the user cannot currently alter the range of the coordinates.

Self-Evaluation

Overall, the required features of the program were successfully implemented and are fully operational. The first extended feature was also implemented and is operational, although it did require a slight redesign of the program and caused the issues outlined above. However, over the course of two and half weeks, the assignment criteria has been met successfully, along with the completion of one extended feature. In addition, the program has the ability to accept multiple configuration files as program arguments. This makes it ideal for processing multiple sets of data at once whilst keeping the data separated appropriately. The assignment therefore provides an appropriate solution to the problem originally outlined in the assignment specification.