**Rust Programming Lab #13 15th November 2022**

**How accurate is your GPS and Histograms**

GPS technology is amazing: you can find your location anywhere on the earth to a few metres. But how accurate is it? In the early days, you could only assume ±100m, but it is, usually, considerably more accurate in 2022. For this assignment, I captured a series of locations from the GPS chip on my laptop: download the test file and you will find a ‘CSV’ (comma separated variables) file containing my observations over two days. If you are curious, you can capture a similar file from the GPS on your laptop or phone. The file has one entry per line:

13.730397, 100.780922,0, 2022-11-14T14:04:59Z,0,0

13.730401, 100.780939,0, 2022-11-14T14:05:11Z,0,2

13.730413, 100.780933,0, 2022-11-14T14:05:22Z,0,1

The first two entries are <latitude>, <longtitude>, followed by some other data (including time), which, for this exercise, you should ignore.

**WARNING:** This is a programming exercise for **YOU**. I want you to show that **you can** build functions to do the needed calculations. It is NOT an exercise in looking up Rust crates to do the work for you. The calculations are fundamentally simple and you should be able to solve the problem directly. Remember, that not every solution to every problem will be found in existing libraries: in real life, you will need to solve many problems that have never been attempted before.

**You must code all the functions yourself. Use of library functions will gain 0 marks.**

**However, for** unpacking the test file into the two angles in the CSV file, you may use a library function – although you solved a very similar problem in a previous lab -adapt or re-use it. It is likely that you will need this task many times in the future, as have many before you, so a host of library routines are available. However searching for library functions to read GPS coordinates is a waste of time, *you can code it yourself simply* – as practice to show that you can solve a similar, but not identical, problem in the future!

1. Create a struct for GPS locations (as before) and an array of these structs.
2. Read the file, line by line, decode the latitude and longtitude, create the structure and add it to an array.
3. Find the median position:  
   Defined as the mean of the latitudes and longtitudes: just sum them separately and find the mean, m, of each. So your mean position will be

mloc = (mlat, mlong)

1. Now calculate the statistical measure known as the **standard deviation, s, for the latitudes and longtitudes.** You can look up a definition of the standard deviation on the web, but basically it an estimate of the error of a set of measurements from a mean. If the distribution of your readings is ‘normal’, then 68% of your measurements will lie within one standard deviation of the mean. Diagram

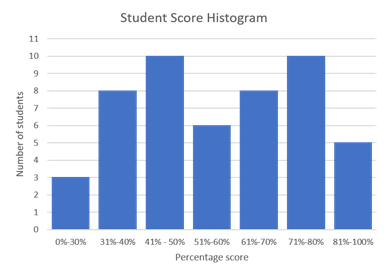
   Description automatically generated
   1. A simple way to calculate, s, for a set of *n* values, *x*j, *j* ∈ {0..n-1} is

where m is the mean,

* 1. Then convert the standard deviation as a pair of angles in latitude and longtitude to **meters**, very approximately, by noting that
     + one degree of latitude ≈ 111,139 m
     + one degree of longtitude ≈ 111,139 cos(latitude), so  
       at Bangkok’s latitude (13.73 N)

one degreee of longtitude ≈ 107963 m

Note this calculation is extremely rough (there is a much more complex 3D angle formula on the web and the earth is not perfectly spherical). However, it will tell you whether you can expect to find your exact parking spot in the carpark or your friend’s front gate 😊 or whether that Ukrainian drone can drop into the escape hatch of that Russian tank. Advanced GPS technology – Differential GPS, etc – is more accurate, but this exercise will tell you how accurately you can expect that commercial GPS chip in your phone to find your friend in the mall, etc.

* 1. Obviously, you should write a set of functions that calculate m, s and the maxima and minima (needed for the histogram in the next step) and also converts the a difference in (slat, slong) to a distance in m.  
     You can either create a set of very simple functions that calculate each value individually, **OR**, note that each metric will need a pass through the same array and that you can calculate them all in a single ‘statistics’ struct that needs only one pass. However, note the simple formula for s needs the mean and two passes anyway. (However, your statistics class – or your Wikipedia class – will tell you how to calculate s without needing the mean first 😊).
  2. Now we should make a **simple histogram** that shows us whether the readings are normally distributed – as in the picture above – or skewed in some way.  
     
  3. Note that a histogram counts the number of occurences in a set of ‘bins’ (i.e. values between 0-30, 31-40, 41-50, .. in the image above).   
     To keep life simple, you might separate the latitude and longtitude values into two separate arrays, then you can trivially, using the same function, compute two histograms – one for latitude variations and one for longtitude variations.
  4. Choose a suitable set of bin values: use the max and min and range = max – min and choose a bin size so that there are about 10 bins. The mean should be in the centre bin. Make an array of the bin limits, e.g. if your range is 0.000093 = max – min = 13.730487 - 13.730394, maybe 10 bins in steps of 0.00001 will work. Your bins would be 13.73039, 13.73040, 13.73041, etc. Check that your mean lies in one of the centre bins.
  5. Fill in the histogram (it will be an array the same size as the number of bins) with counts for the number of values in each bin, e.g. count the number of values - 13.73039 < x <= 13.73040 for the first bin.
  6. Finally, you should print out your histogram so that it is **trivially printed**, e.g. it might look like this, where the number of ‘\*’ is proportional to the count in each bin.

13.73039 \*\*\*\*

13.73040 \*\*\*\*\*\*\*\*\*\*

13.73041 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* 1. Comment on the shape of your plot in your report. Maybe it should like the image for the standard deviation above?

**Answer the questions on the attendance sheet and have a TA sign it off.**

**This lab will run over two weeks, Nov 15 and Nov 22.** The basic statistics are straightforward, you should complete in the first week. The histogram will take more design effort (and some trials to get it right), you should complete it in the following week, Nov 22, **and submit the final report with code on Nov 23.**

**Part of this lab will be marked from results on your attendance sheet. The TAs will confirm that your code was acceptable. Read the error codes on the web site *again*.**

**Then submit a short report AND your code for final assessment for this lab. Put your code in a separate file. *😊***

**Use EduGo. Submit by Nov 23, 23:59.**

**Website: kris.kmitl.ac.th/clinic/Courses/Rust/**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attendance** | **01286120** | **Elementary Systems Programming** | **1 Nov 2022** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name (Thai script\*)** |  | **Student ID**  **65011**   |  |  |  | | --- | --- | --- | |  |  |  | |
| **(Latin characters -  as you enrolled)** |  |
| **\****Please write clearly: practice for one farang who is trying to improve* **😉** | | |

|  |  |  |
| --- | --- | --- |
| **Ex 1** | List structures (e.g. arrays, vectors, structs) used to calculate basic statistics. | **TA** |
|  | Statistics  Latitude Longtitude  Mean, m  Min  Max  SD, s | **TA** |
| **Ex 2** | Estimate of GPS error (calc from s) in m | **TA** |
| **Ex 3** | Histogram bin step size | **TA**  Trait  Prints OK |
| **Ex 4** | *Show the computed histograms in your report.* | **TA** |
|  | Complete a short report and submit your report and code by Nov 15. |  |