**Rust Programming Lab #3 23th August 2022**

**Triangles**

**Exercise 1**

Geometric problems are a good use for structures. In the sample code, you will see that triangles can be modelled by a structure (**struct Triangle**) that has a name (for convenient labelling) and an array of three points. In turn, points are modelled by another structure (**struct Point**). Thus complex structures can be modelled by combinations of smaller ones. The sample code demonstrates one way to read a file containing a set of triangles (**fn read\_triangles**). You do not need to understand the code in **read\_triangles** .. yet 😊 .. we will learn more about it in lectures.

For this assignment, download the sample code and save it as main.rs in your working directory – *xxx*/**src**. You should modify the **check\_triangle** fn to compute some properties of the loaded triangles. You will probably find it helpful to add some simple functions that calculate the lengths of sides, the angles of the triangle and its area. A function that computes the distance between points, **length(d:[Point;2])->f64** in the sample as a guide. Note that the ‘expression’ at the end of the block **{… dd.sqrt() }** is the function return value or you may use an explicit **return dd.sqrt()**.

You may find it convenient to add another function that takes two distinct arguments, *e.g.* **length2(a:Point,b:Point)->f64** as well as functions that calculate the angle between two sides, e.g. **fn angle3( a:Point, b:Point, c:Point ) -> f64**. In turn, angle3 might be simpler if you define a structure, e.g. **struct GVector { .. }** to create a (geometric) vector. It might be a good idea to call it **GVector** to distinguish between mathematical vectors – arrays or **Vec** structures (coming real soon 😊 .. but there are some examples in **read\_triangles** to give you an advance view!).

For checking your code, there is a very convenient way to print out the whole of a structure:

**println!(“Triangle {:?}”,t)**

It is used at the start of **check\_triangle**, you can use “**{:?}**” to print other **struct**’s too!

Finally, add a function to calculate the area of a triangle. There are several ways to calculate a triangle’s area – you may use any convenient one.

For each triangle read from the input file, print out its points, the set of angles and lengths between the sides and the area.

Show your code to the TA and run your program to produce some reasonably readable output (*i.e.* comment out any debugging output first – include debug output from **read\_triangles** 😊), so that the TA can sign off your answers.

**Fill in a sample of your results in the attendance record and hand it in.**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Attendance** | **01286120** | **Elementary Systems Programming** | **23 Aug 2022** |

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

|  |  |  |
| --- | --- | --- |
| **Check 1** | Length of side | TA |
|  | |
| **Check 2** | Angle between sides | TA |
|  | |
| **Check 3** | Additional structs | TA |
|  | |
| **Check 4** | Area | TA |
|  | |
| **Check 5** Output complete and intelligible | | TA |