Segundo Proyecto: Rústico

75.59 - Técnicas de Programación Concurrente I

Objetivo

El objetivo de este proyecto consiste en desarrollar una aplicación que simulare un grupo de jugadores jugando a un juego de cartas denominado "Rústico".

Requerimientos Funcionales

El juego está integrado por **j** jugadores y un coordinador que se dedica a contabilizar los puntos de cada ronda y a establecer la forma de juego en cada ronda: si es *rústica* o no.

Al comenzar el juego se mezcla las cartas de la baraja francesa[1] y se reparte la totalidad de las cartas entre todos los jugadores, quienes deberán colocarlas formando un pilón boca abajo. Si la cantidad de cartas no es múltiplo del número de jugadores, ese remanente se deja sin repartir y el coordinador toma nota.

En cada ronda, se juega una carta del pilón.

Al iniciar cada ronda, el coordinador decide de forma aleatoria y sorpresiva para los participantes la forma de juego: rústica o normal.

Si la ronda es rústica, a partir del momento en el que el coordinador lo anuncia, todos los jugadores pueden apoyar su carta en el pilón central, sin esperar por su turno. Si la ronda es normal, cada jugador debe esperar a su turno, que es en el sentido de las agujas del reloj.

Una vez que todos colocaron su carta, el coordinador pone orden en la mesa y reparte los puntos según las siguientes reglas:

1. El jugador que colocó la carta con el número mayor, recibe 10 puntos. Si hay empate, se dividen los puntos entre los jugadores que empatan.
2. Si, además, la ronda fue rústica:
   - el jugador que apoyó la carta en último lugar pierde 5 puntos **y no puede jugar la ronda siguiente**.
   - el jugador que apoyó la carta en primer lugar gana un punto.

Se juega hasta que se terminen las cartas repartidas a alguno de los jugadores. Resulta ganador el jugador que obtuvo más puntos.

El siguiente es un requerimiento adicional que debe cumplirse:

- La cantidad **j** de jugadores que participan es un número par mayor o igual que 4, y es configurable sin necesidad de recompilar el código.

[1] Las que se usan para jugar al póker.

Requerimientos no Funcionales

Los siguientes son los requerimientos no funcionales de la aplicación:

1. El proyecto deberá ser desarrollado en lenguaje **Rust** utilizando Threads y las siguientes herramientas de concurrencia provistas por el lenguaje y la biblioteca standard *sync*[2]:

   a) Channels (de tipo *multiple producer, single consumer*).

   b) RwLock

   c) Mutex

   d) Barrier (*https://doc.rust-lang.org/std/sync/struct.Barrier.html*)

   e) Condvar (*https://doc.rust-lang.org/std/sync/struct.Condvar.html*)

   El uso de un módulo externo (crate) debe ser previamente autorizado.

   No se permite el uso de crates de concurrencia.

   El código fuente debe compilarse en la versión *stable* del compilador y no se permite utilizar bloques *unsafe*.

2. La simulación no deberá tener interfaz gráfica y se ejecutará en una sola consola de línea de comandos.

3. El proyecto deberá funcionar en ambiente Unix / Linux.

4. La aplicación deberá funcionar en una única computadora.

5. El programa deberá poder ejecutarse en "modo debug", lo cual dejará registro de la actividad que realiza en un único archivo de texto para su revisión posterior.

Tareas a Realizar

A continuación se listan las tareas a realizar para completar el desarrollo del proyecto:

1. Dividir el proyecto en *threads*. El objetivo es lograr que la simulación esté conformada por un conjunto de hilos de ejecución que sean lo más sencillos posible.

2. Una vez obtenida la división en *threads*, establecer un esquema de comunicación entre ellos teniendo en cuenta los requerimientos de la aplicación. ¿Qué *threads* se comunican entre sí? ¿Qué datos necesitan compartir para poder trabajar?

3. Realizar la codificación de la aplicación. El código fuente debe estar documentado.

4. Implementar tests unitarios de las funciones que considere relevantes.

Entrega

La resolución del presente proyecto es individual.
La entrega del proyecto comprende lo siguiente:

1. Informe, se deberá presentar impreso en una carpeta o folio (si las actividades permiten la corrección presencial) y en forma digital (PDF) enviado por correo electrónico a la dirección: *pdeymon@fi.uba.ar*

2. El código fuente de la aplicación, que se entregará únicamente por e-mail. El código fuente debe estar estructurado en un proyecto de *cargo*, y se debe omitir el directorio *target/* en la entrega.

[2] https://doc.rust-lang.org/std/sync/index.html

La fecha de entrega es el miércoles 22 de julio hasta las 19 hs.

El informe a entregar debe contener los siguientes items:

1. Detalle de resolución de la lista de tareas anterior.
2. Listado y descripción de las "*User stories*".
3. Diagrama que refleje los threads, el flujo de comunicación entre ellos y los datos que intercambian.
4. Diagramas de entidades realizados.
5. Diagrama de transición de estados de un jugador.