Adivina el número

Ejercicio N°3

Objetivos	● Diseño y construcción de sistemas con acceso distribuido ● Encapsulación de Threads y Sockets en Clases ● Definición de protocolos de comunicación ● Protección de los recursos compartidos ● Uso de buenas prácticas de programación en C++
Instancias de Entrega	**Entrega 1:** clase 8 (26/05/2020). **Entrega 2:** clase 10 (09/06/2020).
Temas de Repaso	● Definición de clases en C++ ● Contenedores de STL ● Excepciones / RAII ● Move Semantics ● Sockets ● Threads
Criterios de Evaluación	● Criterios de ejercicios anteriores ● Eficiencia del protocolo de comunicaciones definido ● Control de paquetes completos en el envío y recepción por Sockets ● Atención de varios clientes de forma simultánea ● Eliminación de clientes desconectados de forma controlada

Introducción

Se desarrollará un aplicativo por consola, con cliente y servidor, para poder jugar al juego de “Adivina el número”. El servidor deberá ser capaz de aceptar y atender múltiples clientes en simultáneo.

Descripción

Reglas del juego

El juego consiste en que una persona (S) debe pensar un número de **3 cifras no repetidas (N)**, entre 100 y 999, y otra persona (C) debe adivinar el número en menos de 10 intentos. Para esto, por cada número que C intenta adivinar, S le contesta de la siguiente forma:

- Si alguna de las cifras que dijo C se encuentra en el número N, pero está en otra posición, se le computará como **1 regular**.
- Si alguna de las cifras del número que propone C concuerda en valor y posición al número N, se le computará como **1 bien.**
- Si ninguna de las cifras del número que propone C se encuentra en el número N, se le computará como **3 mal.**

El juego termina cuando C adivine el número, habiendo este ganado, o cuando hayan pasado 10 intentos, computándose como derrota la partida.

Formato de Línea de Comandos

Servidor

```
./server <puerto/servicio> <números>
```

Donde `<puerto/servicio>` es el puerto TCP (o servicio) el cual el servidor deberá escuchar las conexiones entrantes.

El parámetro `<números>` representa la ruta a un archivo con una lista de números que el servidor recorrerá en forma **round robin** y usará para que sea el número a adivinar por el cliente en cada partida. El servidor deberá leerlo al inicio y obtener la información relevante. Esta lista se encuentra representada por distintos números separados por un caracter nueva línea (**\n**).

Un ejemplo de este archivo puede ser el siguiente:

```
748
928
903
```

Cuando el primer cliente se conecte, el servidor usará el número 748 para que el cliente lo adivine. Luego usará el número 928 cuando se conecte alguien nuevo, y así sucesivamente, recorriendo en anillo los valores (el siguiente conectado deberá adivinar el número 903, y cuando haya un cuarto cliente conectado, se repetirá el número 748).

Cliente

El cliente se ejecutará de la siguiente forma:

```
./client <ip/hostname> <port/service>
```

El cliente se conectará al servidor corriendo en la máquina con dirección IP `<ip>` (o `<hostname>`), en el puerto (o servicio) TCP `<puerto/servicio>`. Luego se quedará escuchando por entrada estándar los comandos a enviarle al servidor.

Códigos de Retorno

El servidor devolverá 0 si su ejecución fue exitosa, o 1 en caso contrario. El cliente deberá devolver siempre 0, imprimiendo por salida estándar los mensajes de errores que se detallan en la sección siguiente.

Entrada y Salida Estándar

Servidor

Entrada estándar

El servidor esperará el caracter `‘q’` por entrada estándar. Cuando lo reciba, el servidor deberá cerrar el socket aceptador, y esperar a que las conexiones se cierren antes de liberar los recursos y retornar.

Salida estándar

Al finalizar la ejecución, el servidor deberá imprimir una estadística de las partidas jugadas, contando la cantidad de victorias y derrotas que tuvo durante su ejecución.

```
Estadísticas:
      Ganadores:  <cantidad de ganadores>
      Perdedores: <cantidad de perdedores>
```

Notar que las 3 líneas se encuentran terminadas con una nueva línea (**\n**), las líneas de ganadores y perdedores tienen una tabulación al inicio, y la de ganadores tiene dos espacios después de los dos puntos, mientras que la de perdedores tiene un espacio.

```
Estadísticas:\n\tGanadores:  <cantidad de ganadores>\n\tPerdedores: <cantidad de perdedores>\n
```

Salida de error

El servidor imprimirá por la salida de error cuando uno de los siguientes errores ocurren:

- Si la cantidad de parámetros es inválida, se imprimirá:

```
Error: argumentos invalidos.
```

- Si alguno de los números de la lista que recibe por parámetro no se encuentra entre los valores 100 y 999, el servidor imprimirá:

```
Error: archivo con números fuera de rango
```

- Si alguno de los números de la lista posee cifras repetidas, se imprimirá:

```
Error: formato de los números inválidos
```

Cliente

Entrada estándar

Por entrada estándar, el cliente recibirá los comandos que deberá enviar al servidor. Estos son:

```
AYUDA
```

```
RENDIRSE
```

```
<un número de 3 cifras>
```

Los detalles de los comandos serán detallados en la sección del protocolo.

Salida estándar

Por salida estándar se imprimirán los mensajes que se reciben del servidor de acuerdo a los comandos enviados. Estos se verán en la sección del protocolo. El cliente también imprimirá por salida estándar algunos mensajes de error, entre ellos, imprimirá:

- Cuando la cantidad de parámetros de comando de líneas no es el correcto, se imprimirá:

```
Error: argumentos invalidos.
```

- Si el comando recibido no es válido, el cliente imprimirá:

```
Error: comando inválido. Escriba AYUDA para obtener ayuda
```

Protocolo

El protocolo se compone de las siguientes reglas:

- Todos los comandos se enviarán como 1 byte, con el caracter ascii del comando (‘h’ para ayuda, ‘s’ para rendirse y ‘n’ para enviar un número)
    - En el caso de que se envíe un número, seguido del comando se enviará un **entero sin signo de 2 bytes en formato *big endian*** que representa el número recibido por entrada estándar.
- Los strings deberán ser enviados primero con un **entero sin signo de 4 bytes, en formato *big endian***, indicando el largo del string, y luego se envían la tira de bytes del string, sin tener en cuenta el caracter de terminación ‘**\0’**

Comandos

Ayuda

Cuando el cliente reciba por entrada estándar el string AYUDA, le enviará el comando ‘h’ al servidor como se dijo anteriormente. El servidor le responderá con un mensaje de ayuda, enviado como un string único, con el siguiente mensaje:

```
Comandos válidos:\n\tAYUDA: despliega la lista de comandos válidos\n\tRENDIRSE: pierde el juego automáticamente\n\tXXX: Número de 3 cifras a ser enviado al servidor para adivinar el número secreto
```

Formateado, el mensaje se vería de la siguiente forma

```
Comandos válidos:
        AYUDA: despliega la lista de comandos válidos
        RENDIRSE: pierde el juego automáticamente
        XXX: Número de 3 cifras a ser enviado al servidor para adivinar el número secreto
```

Este comando **no** gasta un intento para adivinar el número.

Rendirse

El cliente enviará el byte de valor ‘s’ (ascii 115, 0x73 en hexa). El servidor le responderá con el string:

```
Perdiste
```

Y finalizará la partida, liberando el cliente los recursos solicitados antes de irse. El servidor aumentará en 1 la cantidad de derrotas que hubo.

Número

Cuando el cliente reciba un número por la entrada estándar, deberá enviar primero un byte con el valor ‘n’ (110 en valor ascii, 0x6E en hexa), y luego enviará el número, en el formato previamente indicado (2 bytes sin signo en big endian).

El servidor procesará el número y le contestará al cliente de la siguiente forma:

- Si tuvo uno o más números regulares, le enviará el string:

```
n regular
```

  Con n la cantidad de números regulares que tuvo en ese intento.

- Si tuvo uno o dos números bien, le enviará:

```
n bien
```

- Si tuvo una combinación de ambas (por ejemplo, 1 bien y 2 regulares), siempre se imprimirá primero los bien, y luego los regulares, separado por coma:

```
1 bien, 2 regular
```

- Si las tres cifras no se encuentran en el número secreto:

```
3 mal
```

- Si el número que ingresó el cliente es exactamente el número secreto, este le contestará:

```
Ganaste
```

  E incrementará en 1 la cantidad de ganadores que hubo durante la ejecución. El cliente se irá de forma ordenada.

- Si pasaron los 10 intentos y el cliente no adivinó el número:

```
Perdiste
```

  Y el servidor incrementará en 1 la cantidad de derrotas.

El servidor también hará chequeos de error cuando recibe un número. En particular, enviará:

- Si el número recibido es menor a 100, o mayor a 999, o si alguna de las cifras se repiten (por ejemplo, se pasa el número 990) se enviará:

```
Número inválido. Debe ser de 3 cifras no repetidas
```

Finalmente, si el cliente recibe por entrada estándar un comando inválido, se imprimirá el mensaje:

```
Error: comando inválido. Escriba AYUDA para obtener ayuda
```

Ejemplos de Ejecución

Siempre dicen que un juego se aprende jugando, así que procedamos a dar unos ejemplos de ejecución:

Ejemplo 1

Ejecutamos el servidor de la siguiente forma:

```
./server 2000 server.list
```

El archivo `server.list` posee los siguientes números:

```
748
928
903
```

Después se ejecuta un cliente:

```
./client localhost 2000
```

Este cliente debe adivinar el número 748 para ganar. Si este recibe por entrada estándar:

```
124
```

El cliente enviará el byte ‘n’, seguido del número 124, en formato de 2 bytes sin signo big endian. El servidor contesta con el string:

```
1 regular
```

Debido a que la cifra 4 se encuentra en el número secreto, pero se encuentra en otra posición. Si el cliente envía ahora el número 123, el servidor le contestará:

```
3 mal
```

Supongamos que el cliente envía ahora el número 847, el servidor le contestará

```
1 bien, 2 regular
```

Luego, el cliente recibe el número 745 y el servidor le contesta:

```
2 bien
```

Finalmente, el cliente recibe el número 748, y el servidor contestará finalmente con

```
Ganaste
```

Si cerraramos el servidor, pasándole por entrada estándar el caracter ‘q’, el servidor imprimirá la siguiente estadística:

```
Estadísticas:
        Ganadores:  1
        Perdedores: 0
```

Ejemplo 2

Supongamos que tenemos el mismo servidor del ejemplo anterior, y no fue cerrado (es decir, el jugador 1 se conectó y adivinó el número 748). Supongamos que mientras se estaba desarrollando esa partida, se conecta otro cliente, y este debe adivinar el número 928. Como es su primera vez, este no sabe qué hacer, y escribe el siguiente comando:

```
asdf
```

Como el cliente no puede reconocer ese comando, se imprime el siguiente mensaje:

```
Error: comando inválido. Escriba AYUDA para obtener ayuda
```

Entonces el usuario escribe el comando AYUDA. El cliente enviará el comando ‘h’, y el servidor le contestará:

```
Comandos válidos:
        AYUDA: despliega la lista de comandos válidos
        RENDIRSE: pierde el juego automáticamente
        XXX: Número de 3 cifras a ser enviado al servidor para adivinar el número secreto
```

Supongamos ahora que el usuario es alguien malicioso, y quiere vulnerar nuestro sistema, entonces escribe un número muy grande (mayor a lo que se puede representar en un entero de 2 bytes):

```
23989384759058723409698223469709234059872340923 6
```

El cliente deberá imprimir el mensaje:

```
Error: comando inválido. Escriba AYUDA para obtener ayuda
```

Y luego intenta mandar el número 9384, este número al poder ser representado como un entero sin signo de 2 bytes, lo envía al servidor, quien lo chequea y le responde de la siguiente forma:

```
Número inválido. Debe ser de 3 cifras no repetidas
```

Luego intenta adivinar el número, y escribe el valor 913, y recibe como respuesta:

```
1 bien
```

Nuevamente, con una mente maliciosa, quiere tratar de hacer trampa, y descubrir cuál de las 3 cifras es la correcta, escribiendo el valor 333. El servidor le contestará:

```
Número inválido. Debe ser de 3 cifras no repetidas
```

Finalmente, el usuario frustrado se rinde

```
RENDIRSE
```

Recibe el mensaje:

```
Perdiste
```

Y cierra la aplicación. Si cerraramos el servidor (considerando que mientras estuvo en ejecución se corrió el ejemplo 1 y el ejemplo 2) este imprimirá:

```
Estadísticas:
        Ganadores:  1
        Perdedores: 1
```

Restricciones

La siguiente es una lista de restricciones técnicas exigidas por el cliente:

1. El sistema debe desarrollarse en ISO C++11.
2. Está prohibido el uso de variables globales.
3. Se deberá aplicar al menos 1 polimorfismo
4. Se deberá sobrecargar el operador **()** en alguna clase (puede ser o no la misma que la del punto 4)

Referencias

[1] std::stringstream http://www.cplusplus.com/reference/sstream/stringstream/stringstream/
[2] std::stoi http://www.cplusplus.com/reference/string/stoi/