PRÁCTICA 1 RPC

Enunciado

La práctica consistirá en trasladar el juego del Sudoku escrito en c++ en una aplicación no distribuida a una distribuida mediante llamadas a procesos remotos (RPC).

```
   123 456 789
 +---+---+---+
1|823|756|941|
2|145|239|678|
3|679|148|235|
 +---+---+---+
4|214|365|789|
5|356|897|124|
6|798|412|356|
 +---+---+---+
7|432|681|597|
8|567|924|813|
9|981|573|462|
 +---+---+---+
```

El juego de Sudoku consiste en un tablero de 9 filas por 9 columnas, dividido en 9 bloques de 3 filas por 3 columnas. Los posibles valores que pueden tener cada posición del tablero son números del 1 al 9 verificando la siguiente regla:

Cada fila, columna y bloque del tablero deberá tener una sola ocurrencia de los números del 1 al 9.

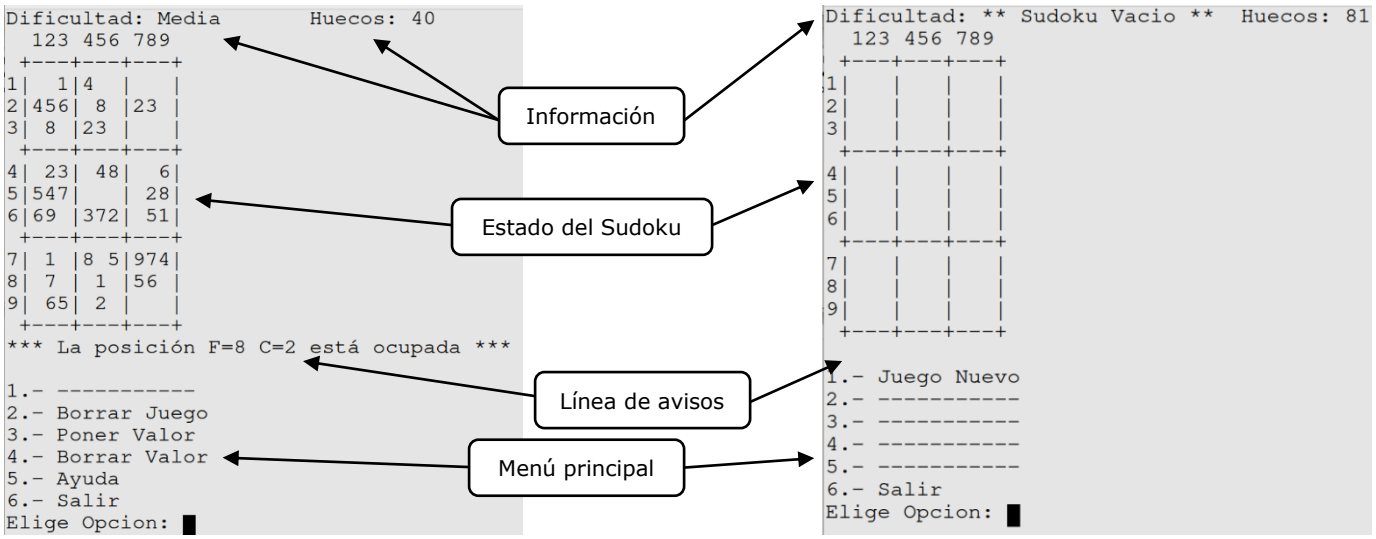
El Juego del sudoku empieza con un tablero con algunos números dispuestos en él y se irán incorporando números en los espacios en blanco del tablero de manera que se verifique en todo momento la regla anterior.

El juego termina cuando todo el tablero se ha rellenado correctamente. El ejemplo mostrado es un sudoku perfectamente terminado.

La complejidad de un sudoku reside en el número de elementos vacíos en el tablero, de manera que a mayor número huecos mayor complejidad.

La aplicación servidor de sudokus podrá almacenar un número ilimitados de Sudokus y para distinguirlos cada sudoku deberá tener un código numérico univoco.

La aplicación cliente mostrará el mismo aspecto que la aplicación sudoku implementada en c++. Dicho aspecto tiene cuatro partes:



Como se puede observar en la imagen, cuando se inicia la aplicación, solo la opción 1 del menú podrá utilizarse para crear un juego y una vez realizado aparecerán el resto de opciones excepto la primera.

Las distintas opciones de menú tienen la siguiente finalidad.

1.- **Juego Nuevo**. Esta opción es la primera que se debe ejecutarse y creará un sudoku con una prioridad solicitada previamente por pantalla.
2.- **Borrar Juego.** Esta opción elimina el juego actual en el servidor, para ello se solicitará previamente una confirmación para realizar la operación.
3.- **Poner un valor.** Solicitará la fila, columna y valor a poner en el tablero. Si ocurre cualquier situación anómala o errónea mostrará un aviso en la línea de avisos de la aplicación.
4.- **Borrar un valor**. Solicitará la fila y la columna del tablero a eliminar. Si ocurre cualquier error mostrará un aviso en la línea de avisos de la aplicación.
5.- **Ayuda.** Pedirá la fila y la columna del tablero y mostrará en la línea de avisos el conjunto de posibles valores que pueden ponerse en dicha posición del tablero.
6.- **Salir.** Desconecta el cliente del servidor. Si se está jugando un partida preguntará si desea eliminar el juego del servidor. Si se confirma borrará el juego mostrando un mensaje de la operación y no se confirma mostrará un mensaje indicando el código de juego que debe utilizarse la próxima vez para recuperar el juego.

Estructuras y tipos de datos

Fichero GestorJuegos.x

```
typedef char Cadena[300];
enum TDificultad {VACIO, MUY_FACIL, FACIL, MEDIA, DIFICIL, MUY_DIFICIL};

struct RCadena
{
   bool Salida; /*TRUE si el campo contenido tiene un sudoku, FALSE si es una cadena vacía.
   Cadena Contenido;
};


struct TFCV  //Estructura para indicar una Fila, Columna y Valor
{
   int pCod;   //Código del sudoku
   int pFil;   //Fila del tablero
   int pCol;   //Columna del tablero
   char pVal;  //Valor que contiene (del 1 al 9 o vacío ' ')
};

struct TFC   //Estructura para indicar una Fila y Columna
{
   int pCod;   //Código del sudoku
   int pFil;   //Fila del tablero
   int pCol;   //Columna del tablero
};


program GESTORJUEGOS {
        version GESTORJUEGOS_VER {
                //Dado una dificultad crea un sudoku y devuelve su código.
                int Nuevo(TDificultad pDifi)=1;

                //Dado un código de sudoku lo elimina. Devuelve TRUE si se ha realizado, FALSE si el juego
                //no existe.
                bool Borrar(int pCod)=2;

                //Actualiza la posición pPos del tablero. Devuelve TRUE si se ha podido actualizar, FALSE
                //si el juego no existe.
                bool PonerValor(TFCV pPos)=3;

                //Devuelve el valor de tablero de la posición pPos. Si el juego no existe devuelve '-'.
                char ObtenerValor(TFC pPos)=4;
```

```
                //Comprueba si la posición pPos indicada puede actualizarse en el tablero. Devuelve TRUE si
                //la posición indicada verifica las reglas del sudoku, FALSE si no las verifica o el juego
                //no existe.
                bool ComprobarValor(TFCV pPos)=5;

                //Devuelve el número de posiciones del tablero que están vacías en el juego con código
                //pCod. Si el juego no existe devuelve el valor -1.
                int NumeroHuecos(int pCod)=6;

                //Verifica que el juego con código pCod no tiene posiciones que incumplen las reglas del
                //sudoku. Devuelve TRUE si el tablero es correcto, FALSE si no lo es o bien si el juego no
                //existe.
                bool Correcto(int pCod)=7;

                //Devuelve una cadena con todos los posible valores que pueden ponerse en la posición pPos.
                //El campo Salida tendrá TRUE si el juego existe o FALSE en caso contrario.
                RCadena Ayuda(TFC pPos)=8;

                //Devuelve una copia formateada del sudoku con código pPos, incluyendo la información del
                //juego y el tablero. Si juego no existe en la información indicará "**Sudoku Vacío **"
                RCadena GetSudoku(int pPos)=9;
        } = 1;
} = 0x30000001;
```

Estructura de datos para almacenar en memoria dinámica los sudokus

```
typedef struct
{
  int Codigo;                 //Código del juego
  char Tablero[9][9];         //Tablero de juego
  TDificultad Dificultad;     //Dificultad
} TJuego;

TJuego *Sudokus=NULL;         //Vector dinámico de Juegos
int NSudokus=0;               //Número de juegos almacenados en el vector Sudokus
```

Fichero de ejemplo

Para esta práctica, se proporciona un código en c++ que implementa el juego del Sudoku en el que se puede basar para realizar esta práctica.

Realización de la práctica en grupo

Esta práctica se puede realizar de manera individual o por grupos. El número de alumnos en cada grupo es dos. Los alumnos que decidan hacerla en grupo deberán notificar al profesor los componentes del grupo y el reparto equitativo de tareas que cada alumno va a realizar. El profesor podrá reorganizar las tareas en caso de que detecte un reparto desigual.