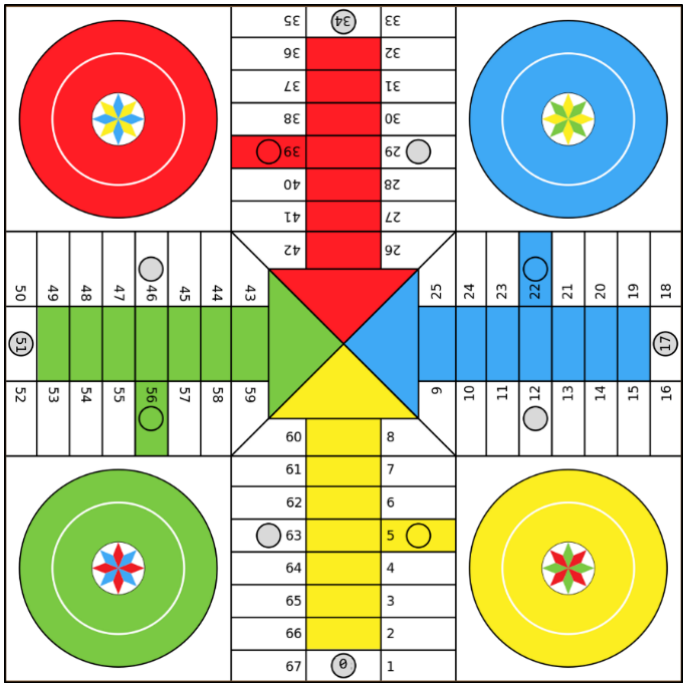
Proyecto *Parchís*

Versión 3

Fecha de entrega: 17 de enero



El objetivo del proyecto es permitir a un usuario humano jugar al parchís contra otros jugadores controlados por la computadora, por medio de un programa en C++ del que ya hemos desarrollado la primera versión. En esta tercera y última versión encapsularemos los datos del juego y, opcionalmente, haremos que tres de los jugadores sean gestionados por la computadora.

La dinámica del juego sigue siendo la misma de la segunda versión.

Obligatoriamente se han de encapsular con estructuras los elementos de datos del juego:

Ahora cada jugador estará representado por una estructura `tJugador` que encapsula la siguiente información:

- ✓ Color del jugador.
- ✓ Fichas del jugador (array de tipo `tFichas`).

Ahora `tJugadores` no es un array de `tFichas`, sino de `NUM_JUGADORES` `tJugador`.

Por otro lado, en lugar de tener dos arrays paralelos para la calle del tablero, ahora declararemos un tipo `tCasilla` para cada casilla, que será una estructura con estos datos:

- ✓ Color de la ficha que hay en esa casilla de la `calle1` (color `Ninguno`, si no hay).
- ✓ Color de la ficha que hay en esa casilla de la `calle2` (color `Ninguno`, si no hay).

Y ahora `tCasillas` será un array de `NUM_CASILLAS` `tCasilla`.

Finalmente tendremos un tipo de estructura `tJuego` encapsulando la información del juego:

- ✓ Casillas del tablero
- ✓ Jugadores
- ✓ Jugador que tiene el turno (color)
- ✓ Tirada actual
- ✓ Premio que se gana
- ✓ Número de seises consecutivos
- ✓ Última ficha movida

1. Cambios en los prototipos (y cabeceras) de los subprogramas

Los cambios en los prototipos y cabeceras de los subprogramas afectan sólo a las listas de parámetros: los distintos parámetros que ahora sean campos de una estructura se sustituirán por un parámetro de tipo de estructura, pasada por valor si no se modifica nada de la estructura y por referencia si se puede modificar algo; y los parámetros que sean arrays paralelos que ahora tienen la forma de un array de estructuras, se sustituirán igualmente por el nuevo tipo de array.

Por ejemplo, si la lista de parámetros es:

```
tJugadores jugadores, tColor& jugadorTurno, tCasillas calle1, tCasillas calle2
```

Esa lista de parámetros se sustituirá por:

```
tJuego& juego
```

Ya que la lista de jugadores, el jugador que juega y las casillas están dentro de una estructura de tipo `tJuego`.

Los arrays paralelos de las dos calles de la versión anterior:

```
tCasillas calle1, tCasillas calle2
```

Se sustituirán por el nuevo array de casillas: `tCasillas casillas`

Se proporciona una nueva versión de `mostrar()`: `void mostrar(const tJuego& juego)`

2. Adaptación de las implementaciones

Los cuerpos de los subprogramas deben adaptarse a la nueva organización de los distintos datos que se acceden. La siguiente tabla muestra algunas correspondencias entre los datos de la versión 2 y los de la versión 3 (aunque algunos nombres puedan ser algo diferentes).

Versión 2	**Versión 3**
`jugadores`	`juego.jugadores`
`premio`	`juego.premio`
`seises`	`juego.seises`

Versión 2	**Versión 3**
`tirada`	`juego.tirada`
`ultimaFichaMovida`	`juego.ultimaFichaMovida`
`jugadorTurno`	`juego.jugadorTurno`
`jugadores[j][f]`	`juego.jugadores[j].fichas[f]`
`calle1[indice]`	`juego.casillas[indice].calle1`
`calle2[indice]`	`juego.casillas[indice].calle2`
`jugadores[jug]`	`juego.jugadores[jug].fichas`

3. El modo depuración

El programa dispondrá de un modo depuración para la ejecución, de forma que estará declarada una constante `Debug` que pueda tener el valor `true` o `false`. Si tiene el valor `true` se cargará la situación inicial del tablero desde un archivo cuyo nombre indique una constante `Archivo` de tipo `string`. Además, las tiradas se leerán también de ese archivo.

Formato del archivo: las primeras 16 líneas contienen la casilla que ocupa cada ficha de los cuatro jugadores (por orden: amarillo, azul, rojo y verde); la siguiente línea contiene el jugador que juega a continuación (0..3); las siguientes líneas contienen los valores de las sucesivas tiradas del dado, terminando con una última línea con el valor -1 como centinela.

```
44
39
-1
-1
65
56
56
-1
25
11
-1
-1
103
36
-1
-1
0
5
5
5
5
5
4
2
1
6
6
-1
```

Casillas de las fichas del jugador amarillo

Casillas de las fichas del jugador azul

Casillas de las fichas del jugador rojo

Casillas de las fichas del jugador verde

Jugador que juega a continuación

Sucesivas tiradas del dado

En el modo de depuración se leerán las tiradas de ese archivo, hasta que se llegue al -1 final, momento en el que se pasará a pedírselas al usuario.

4. Parte opcional: Jugadores manejados por la computadora

Opcionalmente se puede añadir al programa la inteligencia necesaria para que tres de los cuatro jugadores sean manejados por la computadora (el usuario elegirá con cuál jugar). Para ello será necesario crear una versión del subprograma `jugar()` que se llame para los jugadores manejados por la computadora. Esa versión deberá decidir qué ficha mover en aquellos casos en los no haya un único movimiento posible. Hará uso de otros subprogramas que harán comprobaciones sobre las fichas que se pueden mover. Por ejemplo:

- `int puedeComer(tJuego& juego, int& casilla)`: comprobará si alguna ficha de las que se pueden mover comería una ficha de otro jugador en la `casilla` de destino; devolverá el índice de la ficha, o -1 si ninguna ficha movible puede comer.
- `int aMeta(tJuego& juego, int& casilla)`: comprobará si alguna ficha puede llegar a la meta; devolverá el índice de la ficha, o -1 si ninguna ficha movible puede llegar.
- `int aSeguro(tJuego& juego, int& casilla)`: comprobará si alguna ficha puede llegar a una casilla segura; devolverá el índice de la ficha, o -1 si ninguna.
- `int huir(tJuego& juego, int& casilla)`: comprobará si alguna ficha tiene fichas enemigas cerca y debería ser movida para huir del peligro; devolverá el índice de la ficha, o -1 si ninguna.
- `int primeraPosible(tJuego& juego, int& casilla)`: devolverá el índice de la primera ficha que se puede mover, preferiblemente que no esté en un seguro.

Puedes pensar en otras posibilidades. La máquina decidirá, según tu criterio, en qué orden explora las distintas posibilidades. Incluso lo puede hacer de distinta forma para cada jugador máquina (y hasta que todos los jugadores estén controlados por la máquina).

5. Entrega del programa

El archivo `.cpp` se entregará por medio de la correspondiente tarea del espacio virtual de la asignatura. Comenzará con un comentario que, además de identificar a los autores, indicará si se ha realizado la parte opcional o no, así como cualquier otra indicación que se considere relevante sobre el desarrollo de la versión final de proyecto.