

# Práctica 1

## *Dominó: versión 3*

Fecha de entrega: **22 Diciembre de 2019**

### 1.Descripción de la funcionalidad de Dominó V3

En la tercera versión vamos a incorporar el juego completo del dominó. Cada partida será jugada por entre 2 y 4 jugadores, de los cuales uno será el jugador humano y el resto serán simulados por tu programa. Tales jugadores máquina jugarán utilizando ciertas estrategias de juego predefinidas que nos permitirán saber unívocamente qué decisión tomarán en cada situación de juego posible.

Además:

- 1.- Nada más iniciar el programa, el jugador humano podrá elegir entre jugar una partida nueva o bien continuar una partida previa cargándola desde fichero (el jugador indicará el nombre del fichero).
- 2.- Si en algún momento el jugador humano opta por abortar una partida (opción "salir" del menú), se le ofrecerá la posibilidad de guardar la partida en un fichero (el jugador indicará el nombre del fichero), para poder reanudarla posteriormente como se indica en el punto anterior.

A continuación se indican otros aspectos funcionales a tener en cuenta.

#### **Inicio de una partida nueva (es decir, si no se carga una partida previa)**

Al iniciar una partida nueva, se permitirá al jugador elegir el número de jugadores entre 2 y 4. Uno será humano y el resto de jugadores serán programados.

Entonces se generará aleatoriamente el pozo con las 28 fichas posibles. Cada jugador robará al azar 7 fichas del pozo. Las fichas robadas del pozo se eliminarán del pozo.

Empezará el turno el jugador que el seis doble, poniéndose dicha ficha en el tablero. Si ninguno lo tiene, entonces empezará el jugador que tiene cinco doble, y así sucesivamente. Si ningún jugador tiene ningún doble entonces se volverá a generar el pozo y a repartir las fichas, hasta que algún jugador pueda por fin comenzar la partida.

#### **Desarrollo del juego**

Antes del turno de cada jugador se mostrará las fichas colocadas en la mesa (tablero) y las fichas de todos los jugadores.

Si le toca el turno a algún jugador máquina se mostrarán sucesivamente las acciones que tome: colocar una ficha (a derecha o a izquierda, según sea); o bien, si no tiene movimientos válidos, robar una ficha del pozo hasta que pueda poner alguna ficha o se agote el pozo.

Por su parte, las opciones entre las que puede elegir el jugador humano en su turno son:

**1) Colocar** una de sus fichas a la **izquierda** del tablero. En este caso, al igual que se hacía en la versión 2, se pregunta cuál de las fichas del jugador se desea colocar a la izquierda del tablero.

Igual que en versiones anteriores sólo se puede colocar una ficha a la izquierda si uno de sus valores coincide con el valor del extremo izquierdo del tablero.

Si se coloca la ficha debe representarse en el tablero en la posición correcta.

La ficha colocada se elimina de la lista de fichas del jugador.

**2) Colocar** una de sus fichas a la **derecha** del tablero. En este caso se pregunta cuál de las fichas del jugador se desea colocar a la derecha del tablero.

Igual que en el caso anterior, sólo se puede colocar una ficha a la derecha si uno de sus valores coincide con el valor del extremo derecho del tablero.

Si se coloca la ficha debe representarse en el tablero en la posición correcta.

La ficha colocada se elimina de la lista de fichas del jugador.

**3) Robar** una ficha aleatoriamente del pozo (que será eliminada del pozo) y añadirla a la lista de fichas del jugador.

No se permite robar si es posible colocar alguna ficha del jugador en el tablero.

**4) Salir** del juego. Esta opción aborta la partida actual. En este caso se preguntará al jugador si desea guardar su partida para continuarla más adelante. En caso afirmativo, el jugador indicará el nombre del archivo en el que desea guardar la partida.

### **Cambio de turno de jugador**

Cuando un jugador coloca una ficha su turno termina y empieza el turno del siguiente jugador.

Si un jugador no puede colocar ninguna de sus fichas, dicho jugador debe robar sucesivamente una ficha del pozo hasta que pueda colocar alguna ficha.

Si no quedan fichas por robar en el pozo y todavía no puede colocar ninguna ficha, entonces el jugador pasa, terminando su turno sin colocar ninguna ficha.

### **Final de la ronda actual**

Una ronda de dominó termina en uno de los dos siguientes casos:

- 1) Si un jugador coloca todas sus fichas.
- 2) Si no quedan fichas para robar en el pozo y ningún jugador puede poner ficha.

Al terminar la ronda el programa debe mostrar qué jugador ha ganado la ronda (si hay ganador) y cuántos puntos tiene cada jugador (la suma de valores de sus fichas, por lo que el ganador, si lo hay, suma 0). Los puntos se acumulan entre rondas sucesivas. En una partida con varias rondas, el objetivo sería acumular el mínimo número de puntos a lo largo de las rondas jugadas.

Al terminar una ronda, el programa debe preguntar si se quiere jugar otra ronda o bien terminar la partida.

### **Estrategias de los jugadores controlados por la máquina**

Se implementarán dos estrategias para los jugadores no humanos a la hora de intentar colocar una ficha:

- 1.- En cada jugada, la estrategia 1 recorrerá por orden las fichas del jugador máquina (por el orden en que se añadieron en su lista de fichas) y pondrá la primera de ellas que pueda colocar.
- 2.- En cada jugada, la estrategia 2 buscará la ficha que pueda ponerse que sume más puntuación entre sus dos valores (en caso de haber varias fichas que empaten con la misma máxima puntuación, se escogerá la primera que se añadió a lista de fichas del jugador máquina).

En ambas estrategias, si la primera ficha escogida pudiera colocarse por ambos extremos, entonces se colocará por el extremo izquierdo.

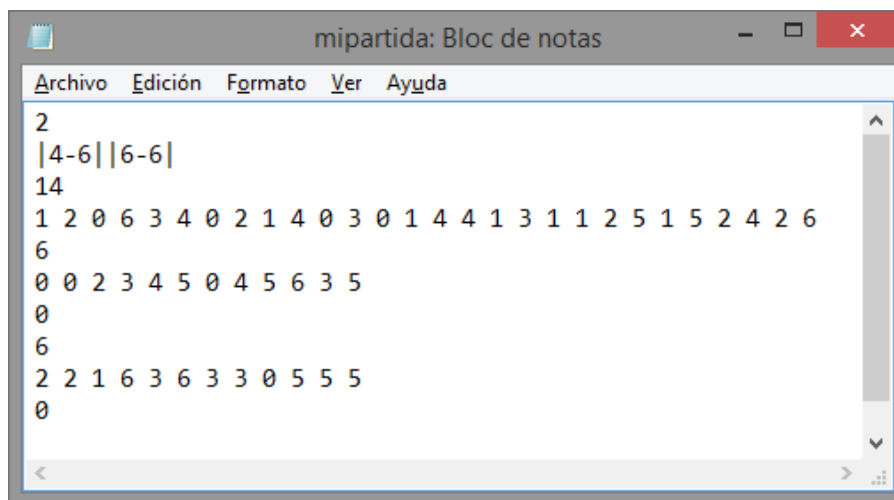
El primer jugador no humano utilizará la estrategia 2, y todos los demás jugadores no humanos sucesivos utilizarán la estrategia 1.

### **Ficheros**

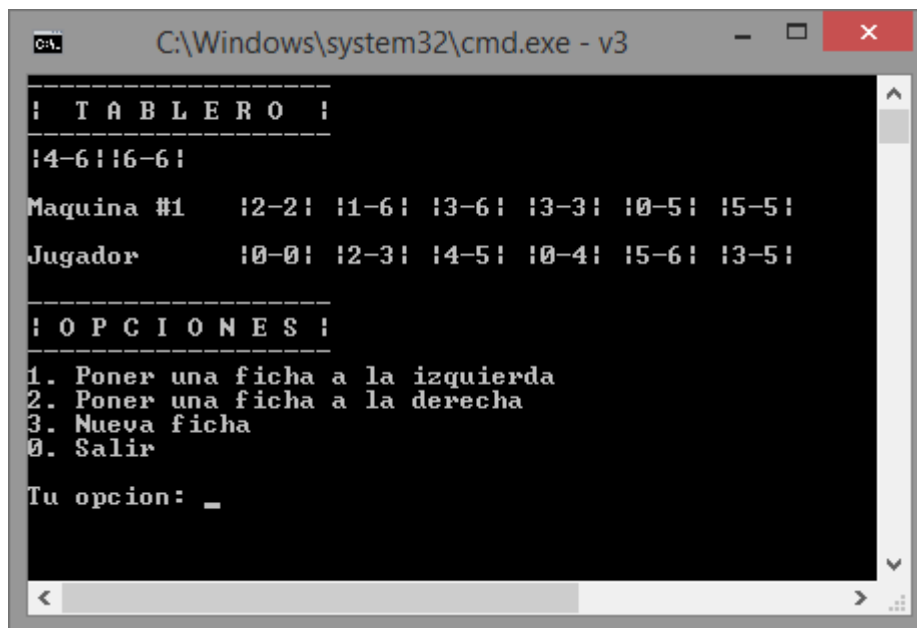
Como se dijo anteriormente, el jugador humano puede abortar la partida que se está jugando (opción "salir" del menú), y volcarla a un fichero para cargarla en otro momento y poderla reanudar. Si se reanuda una partida guardada en un fichero el jugador que tendrá el turno será el que lo tenía cuando la partida se abortó, es decir, el jugador humano.

El formato de los ficheros será el siguiente. En una primera línea encontraremos el número de jugadores. La segunda línea contendrá la representación completa del tablero en formato string. Después encontraremos una línea con el número de fichas en el pozo y otra línea con las fichas del pozo representadas como números separados por espacios, donde cada dos números consecutivos denotarán una ficha. Finalmente, por cada jugador nos encontraremos con tres líneas: la primera indicará el número de fichas que tiene, la segunda contendrá las fichas en sí mismas (siguiendo el mismo formato que indicamos para las fichas del pozo), y la tercera nos dirá la puntuación acumulada por dicho jugador durante las rondas anteriores de la partida. El primer grupo de tres líneas de un jugador corresponderán al jugador humano; las restantes, a las sucesivas máquinas que jugaban la partida.

Un posible fichero sería el siguiente, que representaría una partida con dos jugadores (humano y máquina) en la que se habría comenzado la primera ronda. En concreto se estaría en el tercer turno de dicha ronda (el primer turno fue para el humano que comenzó poniendo su seis doble, el segundo turno fue para la máquina que puso la otra ficha del tablero, y el tercer turno es para el humano que abortó la partida en dicho turno).



Si cargásemos la partida guardada en ese archivo, el estado del juego sería el que figura a continuación:



```
C:\Windows\system32\cmd.exe - v3

! T A B L E R O !
!4-6!!6-6!
Maquina #1  !2-2! !1-6! !3-6! !3-3! !0-5! !5-5!
Jugador     !0-0! !2-3! !4-5! !0-4! !5-6! !3-5!

! O P C I O N E S !
1. Poner una ficha a la izquierda
2. Poner una ficha a la derecha
3. Nueva ficha
0. Salir
Tu opcion: _
```

Fíjate en que primero se visualizan los jugadores máquina y el jugador humano se visualiza el último.

## 2. Detalles de implementación de Dominó V3

En la versión 3, como ya hiciste en la versión 2, deberás definir una constante `NumFichas` para el tamaño de los arrays donde se guardarán las fichas. También definirás otras dos constantes para guardar los números máximo y mínimo permitidos de jugadores (`MinJugadores`, `MaxJugadores`).

Además, usa structs, arrays o combinaciones de ambos para representar los siguientes tipos:

- `tFicha` para representar fichas, conteniendo sus dos valores numéricos.
- `tArrayFichas` para representar arrays de fichas.
- `tListaFichas` para representar listas de fichas, con su array y su contador.
- `tJugadores` para representar arrays de jugadores, es decir, arrays de listas de fichas. La primera componente del array guardará la lista de fichas del jugador humano; las siguientes guardarán las de las máquinas.
- `tPuntosJugadores` para representar arrays que contendrán la puntuación numérica acumulada durante las rondas anteriores por todos los jugadores. La primera

componente del array guardará los puntos acumulados por el jugador humano; las siguientes, los de las máquinas.

- `tJuego` para representar el juego, que contendrá la siguiente información: el pozo, el array de jugadores, el array de puntuaciones de los jugadores y el número de jugadores.

La utilización de los tipos anteriores te obligará a cambiar muchos de los subprogramas desarrollados en la versión anterior. Afortunadamente, esto simplificará sus cabeceras pues reducirá su número de parámetros: los tipos que se definen en la versión 3 permitirán que la información necesaria para cada subprograma sea transportada utilizando menos variables, de manera más compacta y legible.

Además de adaptar los subprogramas de la versión anterior a los nuevos tipos, deberás crear otros para desarrollar las nuevas funcionalidades de la versión 3, entre los que se encontrarán, al menos, los siguientes:

`bool leerJuego(tJuego& juego)`: solicita el nombre de un archivo que contiene una partida y la lee y devuelve en `juego`. Si la lectura no se puede realizar devolverá `false`; en caso contrario devolverá `true`.

`void leerListaFichas(ifstream& entrada, tListaFichas& listaFichas)`: lee y devuelve las fichas de la siguiente línea del archivo de entrada. Este subprograma se usará en `leerJuego`.

`void escribirJuego(const tJuego& juego)`: solicita el nombre de un archivo en el que guardar el estado de la partida representada en `juego` y la almacena según el formato descrito anteriormente en el enunciado.

`void escribirListaFichas(ofstream& salida, const tListaFichas& listaFichas)`: escribe las fichas de la lista dada en la siguiente línea del archivo de salida. Este subprograma se usará en `escribirJuego`.

`int quienEmpieza(const tJuego& juego, int& indice)`: devuelve el jugador que arranca la partida (0: humano, >0: máquina correspondiente, -1: nadie) y en `indice` devuelve la posición (en la lista de fichas del jugador) de la ficha con la que se arrancará.

`void iniciar(tJuego& juego, int& jugador)`: partiendo de un juego sin inicializar crea la configuración inicial de la partida. Para ello crea el pozo, lo desordena, roba 7 fichas para cada jugador, y localiza el jugador que puede arrancar

la partida colocando la primera ficha en el tablero (el que tiene el mayor doble). Si nadie puede poner la primera ficha (nadie tiene un doble), se vuelven a repetir las acciones anteriores hasta conseguirlo. Una vez conseguido se coloca la ficha (el mayor doble) en el tablero y en `jugador` se devuelve el jugador (0: humano, >0: máquina correspondiente) al que le corresponderá el primer turno.

`bool sinSalida(const tJuego& juego)`: devuelve un valor booleano que indica si se ha producido o no un bloqueo del juego (el pozo se ha quedado sin fichas y además ningún jugador puede colocar ninguna de sus fichas).

`bool estrategia1(tJuego& juego, int jugador)`: comprueba si el jugador máquina correspondiente puede realizar un movimiento aplicando la estrategia 1, y si es posible lo hace; devuelve si logró realizar algún movimiento.

`bool estrategia2(tJuego& juego, int jugador)`: comprueba si el jugador máquina correspondiente puede realizar un movimiento aplicando la estrategia 2, y si es posible lo hace; devuelve si logró realizar algún movimiento.

## Entrega de la práctica

La práctica se entregará en el Campus Virtual por medio de la tarea **Entrega de la versión 3 de la práctica**, que permitirá subir el archivo *main.cpp* con el código fuente.

Uno de los dos miembros del grupo será el encargado de subirlo, no es necesario que lo suban los dos.

Recordad poner el nombre de los miembros del grupo en un comentario al principio del archivo de código fuente.