Práctica 3 FP1 2019
Dominó V3

Fecha de entrega: 15 de 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 el jugador 1 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.- Al iniciar el programa, el tablero aparecerá vacío y el usuario podrá elegir entre jugar una partida nueva o bien continuar una partida previa, cargándola desde fichero (el usuario podrá indicar el nombre del fichero).

2.- En cualquier momento al jugador se le ofrecerá la posibilidad de guardar la partida para poder reanudarla posteriormente como se indica en el punto anterior (el usuario podrá indicar el nombre del fichero).

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. El primer jugador será humano, y el resto de jugadores serán programados.

Entonces se genera aleatoriamente el pozo con todas las fichas posibles. Tendremos 28 fichas siendo los valores de las fichas de 0 a 6. Cada jugador roba al azar 7 fichas del pozo. Las fichas robadas del pozo se eliminan.

Empieza el turno el jugador que tiene el doble del máximo valor posible (el seis doble si los valores de las fichas llegan hasta 6), poniéndose dicha ficha en el tablero. Si ninguno lo tiene, entonces empieza el jugador que tiene el doble inmediatamente inferior (cinco doble en el ejemplo anterior), y así sucesivamente. Si ningún jugador tiene ningún doble entonces se vuelven a repartir las fichas, hasta que algún jugador pueda por fin comenzar la partida.

Desarrollo del juego:

Antes del turno de cada jugador, se muestran las fichas colocadas en la mesa (tablero) y las fichas de todos los jugadores. Debes añadir una constante en tu programa para controlar si se muestran las fichas tanto del humano como de los jugadores controlados por la máquina, o bien si se muestran solo las fichas del humano, en cuyo caso sólo se indica el número de fichas que tiene cada uno de los jugadores máquina (pero no cuáles). Si le toca el turno a algún jugador máquina, se mostrarán sucesivamente las acciones que tome: colocar una ficha, o bien robar una ficha del pozo si no tiene movimientos válidos, hasta que pueda poner alguna ficha o se agote el pozo.

Las opciones que puede elegir el jugador humano en su turno son:

1) Colocar una de sus fichas a la **izquierda** del tablero. En este caso se pregunta cuál de las fichas del jugador se desea colocar a la izquierda del tablero. El jugador lo indicará con un número (1 será su primera ficha, y así sucesivamente).

Sólo se puede colocar una ficha si uno de sus valores coincide con el valor del extremo izquierdo del tablero.

Si se coloca la ficha, entonces 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. Se actúa similarmente al caso anterior.

3) Robar una ficha aleatoriamente del pozo que se elimina de éste y se añade a las fichas del jugador (se saca la última ficha del pozo y se añade como última ficha del jugador).

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

4) Guardar la **partida** para continuarla más adelante. El usuario indicará el nombre del archivo en el que desea guardar su partida.

5) Salir del juego. En este caso se pedirá al usuario confirmación para abandonar el juego.

Cambio de turno de jugador:

Si un jugador coloca correctamente 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 la última 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 el turno del jugador 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:

1.- En cada jugada, la estrategia 1 recorrerá por orden las fichas del jugador (por el orden en que se añadieron a las fichas del jugador) 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 haber varias fichas que empaten con la misma máxima puntuación, se escogerá la primera que se añadió a las fichas del jugador).

En ambos casos, 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, la partida puede ser salvada/cargada a un fichero para continuar otro día. El formato de los ficheros será el siguiente. En un primer renglón encontraremos el número de jugadores y el valor máximo de las fichas. El segundo renglón contendrá la representación completa del tablero en formato string. Después encontraremos un renglón con el número de fichas en el pozo y otro renglón 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 renglones: el primero indicará el número de fichas que tiene, el segundo denotará las fichas en sí mismas (siguiendo el mismo formato que indicamos para el pozo), y el tercero nos dirá la puntuación acumulada por dicho jugador durante las rondas anteriores de la partida. Un posible fichero sería el siguiente, que denotaría una partida en la que acabamos de comenzar la primera ronda (el renglón que comienza con 7 8 termina debajo, con 0 3):

```
4 9
|3-7||7-7|
27
7 8 1 4 8 8 3 4 2 7 2 5 0 1 3 9 4 6 6 9 9 9 5 5 7 9 0 5 0 8 4 9 0 6
4 7 4 5 0 4 3 6 2 4 0 7 1 9 4 4 4 8 0 3
7
0 2 6 7 5 7 1 5 1 8 0 9 0 0
0
7
3 5 5 9 1 7 6 6 1 1 2 9 1 6
0
6
6 8 3 8 2 3 3 3 2 6 5 8
0
6
1 2 1 3 2 2 5 6 8 9 2 8
0
```

La Figura 1 muestra la ejecución del programa tras cargar los datos del fichero anterior:

```
------------------
|  T A B L E R O  |
------------------
|3-7||7-7|

Máquina #1   |3-5| |5-9| |1-7| |6-6| |1-1| |2-9| |1-6|
Máquina #2   |6-8| |3-8| |2-3| |3-3| |2-6| |5-8|
Máquina #3   |1-2| |1-3| |2-2| |5-6| |8-9| |2-8|

Jugador      |0-2| |6-7| |5-7| |1-5| |1-8| |0-9| |0-0|
```

Figura 1: Ejemplo de ejecución

2. Detalles de implementación de la versión 3

En la versión 3 deberás definir una constante para indicar si se ocultarán las fichas de los jugadores máquina o no (`FICHAS_OCULTAS`), así como otras constantes que indicarán el tamaño de los arrays que usarás para guardar listas de fichas (`NUM_FICHAS`), y los números máximos y mínimos permitidos de jugadores y de valores utilizados en las fichas (`MIN_JUGADORES, MAX_JUGADORES, MIN_PTOS_FICHA, MAX_PTOS_FICHA`).

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

- `tFicha` para representar las fichas, conteniendo sus dos valores numéricos.

- `tArrayFichas` para representar arrays de fichas.

- `tListaFichas` para representar listas de fichas, con su array y su contador.

- `tJugador` para representar la lista de las fichas del jugador y sus puntos, es decir, la puntuación numérica acumulada durante las rondas jugadas por el jugador.

- `tArrayJugadores` para representar el array de jugadores de la partida.

- `tJuego` para el juego, que contendrá la siguiente información: el número de jugadores, el máximo valor permitido en las fichas, el tablero, el pozo, las fichas de los jugadores y sus puntuaciones.

La utilización de los tipos anteriores te obligará a cambiar muchas de las funciones desarrolladas en la versión anterior. Afortunadamente, esto simplificará muchas de sus cabeceras, pues reducirá su número de parámetros: los tipos anteriores permitirán que la información necesaria para cada función sea transportada utilizando menos variables, de manera más compacta y legible.

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

```
//Guardar la partida:
void guardarJuego(const tJuego juego);

//Guarda las fichas en el siguiente renglón del archivo de salida:
void guardarListaFichas(ofstream salida, const tListaFichas
listaFichas);

//Carga la partida:
bool cargarJuego(tJuego juego);

//Carga las fichas del siguiente renglón del archivo de entrada:
void cargarListaFichas(ifstream entrada, tListaFichas listaFichas);
```

```
//Devuelve quién empieza (0: humano, >0: máquina, -1: nadie) y, en
el índice, la posición de la ficha con la que empezar:
int quienEmpieza(const tJuego juego, short int indice);

//Partiendo de un juego sin inicializar, crea la configuración
inicial de la partida, en la que ya se ha colocado la primera pieza
(del jugador que tenía el mayor doble) en el tablero; si nadie lo
tiene, se repetirá el reparto hasta que se consiga. En el parámetro
jugador se devuelve el jugador al que le toca colocar ficha:
void inicializarJuego(tJuego juego, short int jugador);

//Indica si el pozo se ha quedado sin piezas y, además, ningún
jugador puede colocar ninguna de sus piezas:
bool sinSalida(const tJuego juego);

//Realiza el movimiento del jugador máquina correspondiente
utilizando la estrategia 1 (y devuelve si logró realizar algún
movimiento):
bool estrategia1(tJuego juego, int jugador);

//Lo propio para la estrategia 2:
bool estrategia2(tJuego juego, int jugador);
```

Parte opcional

1) Al iniciar una partida el jugador podrá escoger el valor numérico máximo de las fichas de juego, entre 6 y 9 (el mínimo será siempre 0). Este valor afectará al número de fichas del juego. De este modo tendremos 28 fichas si los valores de las fichas son de 0 a 6, y 55 fichas si los valores fueran de 0 a 9.

2) Modifica tu programa para permitir partidas donde todos los jugadores son jugadores máquina, es decir, donde ningún jugador es humano. Para ello, añade a tu programa una constante que controle si el programa se comportará de la manera normal, o bien si una máquina sustituirá siempre al jugador humano (tanto cuando se inicia una partida nueva como cuando se carga una partida desde fichero).

3) La máquina que sustituye al humano llevará a cabo una tercera estrategia más sofisticada que las anteriores. Dicha máquina mirará las fichas que tiene y las fichas que están puestas en el tablero (pero no las fichas que tienen los demás jugadores humano o máquina o que están en el pozo, que no puede conocer) para contar cuántas fichas *que no tiene él ni están en el tablero* contienen cada número posible desde el 0 hasta el máximo permitido en la partida actual. Entonces, de entre todas las fichas que pueda poner en su jugada, la máquina escogerá aquella que deje, como nuevo extremo izquierdo o derecho, algún número N que contenga alguna de sus

otras fichas (para que él mismo pueda seguir en el próximo turno por dicho extremo, si ningún otro jugador se le adelanta extendiendo el tablero por ahí) pero, *además*, se minimice el número de fichas que no tiene él ni están en el tablero que contienen dicho número N (para que sea más improbable que algún rival pueda "pisarle" dicha posible futura jugada).

Si hay varias jugadas posibles que cumplen dicho criterio y empatan en dicha minimización, entonces se desempatará como se hacía para las estrategias 1 y 2: primero la ficha más antigua posible, y si tal ficha se puede añadir por la izquierda y por la derecha con el mismo valor en la minimización anterior, se añadirá por la izquierda. Si no hay ninguna jugada que cumpla tal criterio, entonces la máquina utilizará la estrategia 2 en su movimiento.

4) Opcionalmente también, puedes crear tu propia estrategia para el último jugador máquina (recordemos que dijimos que el primer jugador máquina usaría la estrategia 2, y los demás la estrategia 1). En tal caso, una nueva constante de tu programa permitirá controlar si dicho último jugador máquina seguirá utilizando la estrategia 1, o bien si utilizará la nueva estrategia que has diseñado para él.

Entrega de la práctica

La práctica se entregará en el Campus Virtual por medio de la tarea **Entrega 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.