Práctica 1
Las siete y media

Fecha de entrega: 9 de diciembre de 2018

Don Mendo:
...Y el de Vedia
Dijo: "No os aburriréis;
os propongo, si queréis,
jugar a las siete y media".

Magdalena:
¿Y por qué marcó esa hora
tan rara? Pudo ser luego…

Don Mendo
Es que tu inocencia ignora
que, a más de una hora, señora,
las siete y media es un juego.

Magdalena
¿Un juego?...

Don Mendo
...Y un juego vil
que no hay que jugarlo a ciegas,
pues juegas cien veces, mil,
y de las mil, ves febril
Que o te pasas o no llegas.
Y el no llegar da dolor,
pues indica que mal tasas
y eres del otro deudor.
Mas ¡ay de ti si te pasas!
¡Si te pasas es peor!

(Extraído de "La venganza de don Mendo", de D. Pedro Muñoz Seca)

1. Descripción de aspectos básicos del juego

En esta práctica tienes que ir desarrollando de manera incremental un programa que permita jugar a distintas variantes, cada vez más perfeccionadas, de *Las siete y media*. En todas nuestras variantes intervendrán dos jugadores en cada partida: el jugador humano y la máquina, en este orden.

En el juego se usa un mazo con las 40 cartas de la baraja española dispuestas de forma aleatoria. En una baraja española las 40 cartas están divididas en cuatro grupos (oros, copas, espadas, bastos) de diez cartas cada uno. Las cartas que tienen asociado un número entre uno y siete se llaman como el número que tienen y las cartas que tienen asociado un diez, un once o un doce se llaman sota, caballo o rey, respectivamente. En nuestro juego cada carta tiene un valor dependiendo del número que tiene asociado, de forma que si el número asociado está entre el uno y el siete, el valor de esa carta es dicho número; y si el número de la carta es diez, once o doce (sota, caballo o rey) el valor de esa carta es un medio.

El objetivo del juego es que, en cada partida, cada uno de los jugadores, a base de robar cartas de forma consecutiva del mazo, consiga acercarse lo más posible a siete y media, sin pasarse. La puntuación obtenida por el jugador en una partida es la suma de los valores de sus cartas. Si con las cartas robadas del mazo un jugador supera los 7,5 puntos, perderá automáticamente la partida y ganará su oponente. Si

ninguno de los dos jugadores se pasa (ninguno supera los 7,5 puntos), ganará el que más se acerque a los 7,5 puntos. Si obtienen el mismo número de puntos, por defecto se elegirá aleatoriamente quién es el ganador.

2. Descripción de la funcionalidad de la aplicación

Versión 1.- En esta primera versión del juego el mazo con las 40 cartas estará en un fichero de texto y se ofrecerá un menú en el que el jugador humano decidirá si juega la partida en el modo A, en el modo B (ambos modos serán explicados más abajo) o si finaliza el programa. El programa actuará de forma cíclica, permitiendo jugar hasta que se elija finalizar el programa. En caso de elegir jugar una partida, ya sea en el modo A o ya sea en el modo B, el jugador humano introducirá el nombre del archivo donde está el mazo de cartas y a continuación se procederá a jugar la partida, volviendo a aparecer el menú cuando la partida haya acabado.

En el modo A cada jugador estará obligado, cuando le llegue el turno, a robar un número concreto de cartas[1]. Ese número de cartas, que será el mismo para los dos jugadores, se generará aleatoriamente al comenzar la partida y estará en el rango [3, 5].

En el modo B, antes de comenzar la partida se generará aleatoriamente el número máximo de cartas, *max_cartas*, en el intervalo [3, 5], que cualquiera de los dos jugadores podrá coger, pero los jugadores no estarán obligados a coger *max_cartas* cartas, sino que se pueden plantar antes si temen que van a pasarse. Así, el primer jugador (humano) robará una primera carta del mazo de cartas y seguirá robando cartas del mazo hasta que decida plantarse (es decir, podrá parar de coger cartas si considera que su puntuación es suficientemente buena), hasta que haya cogido *max_cartas* cartas o si se pasa. Cuando llegue el turno de la máquina ésta hará lo mismo, pero como se tratará de una máquina muy simple (¡y a la que le gusta asumir riesgos!), como regla de decisión para plantarse simplemente comprobará si su puntuación supera la del humano.

En ambos modos, para cada jugador se mostrará por pantalla cada carta robada y para cada uno de ellos se mantendrá en memoria únicamente la puntuación que consiga, puntuación que se irá mostrando actualizada tras cada robo de carta. Si el jugador humano se pasa, automáticamente finalizará la partida; lo mismo ocurrirá si se pasa la máquina.

[1] En todos los modos de juego que usan un archivo como contenedor del mazo de cartas, éstas se van robando en el orden en que figuran en el archivo, y la máquina empieza cogiendo la carta siguiente a la última robada por el humano.

Versión 2.- En esta segunda versión el menú permitirá seleccionar un tercer modo de juego (modo C). Las diferencias en este modo de juego frente a los anteriores son:

1. Ninguno de los jugadores tendrá limitación en el número de cartas que puedan coger (más allá de las que existan en el mazo en cada momento).
2. La máquina se ha vuelto más prudente y se plantará cuando su puntuación sea superior a la del humano o, en caso de que estén empatados, si la probabilidad de pasarse supera el 50%.

En esta segunda versión será necesario llevar un recuento de cuántas cartas de cada tipo quedan en el mazo, de forma que la máquina pueda estimar cómo de probable es que se pase.

Supongamos una partida en la que el humano ha robado tres cartas y se planta (p.e., un 3, un 2 y una figura, lo que hace que el humano tenga una puntuación de 5,5). En ese momento el recuento de cartas que quedan en el mazo es:

Figura	Uno	dos	tres	cuatro	cinco	seis	siete
11	4	3	3	4	4	4	4

Llega el turno de la máquina. Imaginemos que la máquina va robando y llega a un estado con cinco cartas robadas (un 3, un 1 y tres figuras, lo que hace que tenga acumulados también 5,5 puntos). En ese momento máquina y humano estarían empatados. Podemos conocer la probabilidad que tendría la máquina de pasarse si robara una carta más (es decir, la probabilidad de robar una carta >= 3) sabiendo que quedan un total de 32 cartas en el mazo y que dichas cartas son:

Figura	Uno	dos	tres	cuatro	cinco	seis	siete
8	3	3	2	4	4	4	4

probabilidad (sacar >= 3) = (2 treses + 4 cuatros + 4 cincos + 4 seises + 4 sietes) / 32 = 0,56

En este primer supuesto la máquina decidiría no arriesgarse y que fuese el azar quién decidiese el ganador.

Supongamos ahora que la partida hubiese ido de otra forma: el humano ha robado un 4 y se ha plantado, y la máquina ha robado y también ha sacado un 4. En este caso el recuento de cartas que quedan en el mazo sería:

Figura	Uno	dos	tres	cuatro	cinco	seis	siete
12	4	4	4	2	4	4	4

Y la probabilidad de que la máquina se pase si roba una carta más sería:

probabilidad (sacar >= 4) = (2 cuatros + 4 cincos + 4 seises + 4 sietes) / 38 = 0,37

En este caso la máquina asumiría el riesgo y robaría, al menos, otra carta.

Versión 3.- En esta tercera y última versión de la práctica se llevará la cuenta del número de partidas que se juegan en la ejecución correspondiente y el menú permitirá seleccionar un cuarto modo de juego (modo D). En este cuarto modo de juego:

1. El mazo de cartas no estará en un archivo, sino que lo generará el programa para cada partida y se mantendrá en memoria. Inicialmente contará con las 40 cartas dispuestas de forma aleatoria, e irá reduciéndose el número de cartas a medida que éstas vayan siendo robadas por los jugadores.
2. Se mantendrá en memoria no sólo la puntuación que va consiguiendo cada jugador, actualizada a medida que roba cartas, sino también las cartas que roba, de forma que cada vez que roba se muestren todas las cartas que tiene en su haber, no sólo la que acaba de robar.
3. El comportamiento del jugador humano y de la máquina serán los mismos que en el modo C.
4. En caso de que se produzca un empate, no se recurrirá por defecto al azar para determinar al ganador, sino que el ganador será el que haya conseguido la puntuación con el menor número de cartas. En caso de haber conseguido idéntica puntuación con idéntico número de cartas, el azar decidirá quién gana.
5. Quién ha sido el ganador, así como la puntuación y las cartas robadas por cada jugador, se almacenarán en un archivo de texto cuyo nombre coincidirá con el del número de la partida.

3. Detalles de implementación

A continuación se muestran las indicaciones que hay que seguir en el desarrollo de las diferentes versiones.

Formato de los archivos de entrada (archivos de cartas) para los modos A, B y C

Los archivos que contienen los mazos con las 40 cartas son archivos de texto con 40 números, representando cada uno de ellos una carta de la baraja. Cada número aparece en una línea. Puesto que se trata de números, no hay información acerca del palo.

Generación de números aleatorios

Para generar números aleatorios debes utilizar las funciones `rand()` y `srand(semilla)` de la biblioteca `cstdlib`. Una secuencia de números aleatorios comienza en un primer número entero que se denomina semilla. Para establecer la semilla el programa deberá invocar a la función `srand` con el argumento deseado. Lo que hace que la secuencia se comporte de forma aleatoria es precisamente la semilla. Una semilla habitual es el valor de la hora del sistema que se obtiene con una invocación a `time(NULL)`, de la biblioteca `ctime`, ya que así es siempre distinta para cada ejecución. Así pues, el programa deberá invocar *una única vez* a `srand(time(NULL))`. Una vez establecida la semilla, la función `rand()` genera, de forma pseudoaleatoria, otro entero positivo a partir del anterior. Si quieres que los números aleatorios generados estén en un determinado intervalo, deberás utilizar el operador `%`. Así, para obtener un entero aleatorio en el intervalo [limiteInferior, limiteSuperior], hay que usar la expresión `limiteInferior + rand() % (limiteSuperior+1-limiteInferior)`.

Requisitos para la primera versión

La versión 1 deberá incluir al menos los siguientes subprogramas:

- `float modoA (ifstream& file, int numCartas)` Permite a cualquiera de los dos jugadores realizar su turno del modo A en una partida. Recibe el archivo con el mazo y el número de cartas que hay que robar, y devuelve los puntos obtenidos tras robar ese número de cartas.
- `float modoBhumano (ifstream& file, int numCartas)` Permite realizar el turno del jugador humano en el modo B. Recibe el archivo con el mazo y el número máximo de cartas que puede robar, y devuelve los puntos obtenidos.
- `float modoBmaquina (ifstream& file, int numCartas, float puntosHumano)` Permite realizar el turno del jugador máquina en el modo B. Recibe el archivo con el mazo, el número máximo de cartas que puede robar y la puntuación obtenida por el jugador humano, y devuelve los puntos obtenidos.
- `int determinaGanador (float puntosJugador, float puntosMaquina)` Recibe los puntos obtenidos por el jugador humano y por

la máquina, y devuelve un valor que indica quién gana (1 = gana el humano, 2 = gana la máquina).

Requisitos para la segunda versión

En la versión 2 deberás incluir un tipo array `tCartasPorAparecer` cuyas variables permitan llevar un recuento de cuántas cartas de cada tipo quedan en el mazo. Además deberás incorporar, al menos, los siguientes subprogramas:

- `void modoChumano (ifstream& file, tCartasPorAparecer cartas, float& puntos)` Permite realizar el turno del jugador humano en el modo C. Recibe el archivo con el mazo y una variable `cartas` que indica cuántas cartas de cada tipo quedan, y devuelve los puntos obtenidos y actualiza `cartas` de acuerdo con las cartas que haya robado el humano.
- `void modoCmaquina (ifstream& file, tCartasPorAparecer cartas, float puntosHumano, float& puntos)` Permite realizar el turno del jugador máquina en el modo C. Recibe el archivo con el mazo, una variable `cartas` que indica cuántas cartas de cada tipo quedan y la puntuación obtenida por el jugador humano, y devuelve los puntos obtenidos y actualiza `cartas` de acuerdo con las cartas que haya robado la máquina.
- `bool esProbablePasarse (float puntosMaquina, const tCartasPorAparecer cartas)` Determina si la probabilidad que tiene la máquina de pasarse si robara una carta más es mayor que 0.5. Recibe la puntuación actual de la máquina y una variable `cartas` que indica cuántas cartas de cada tipo quedan, y devuelve `true` si la probabilidad de pasarse si roba una carta más supera 0.5 y `false` en caso contrario.

Formato de los archivos de salida (archivos de resultados) del modo D

El archivo de resultados que se genera al jugar una partida en el modo D contendrá 4 líneas: la primera con el número de partida, la segunda con el ganador, la tercera con la puntuación y las cartas del humano, y la cuarta con la puntuación y las cartas de la máquina.

Requisitos para la tercera versión

En la versión 3 deberás incluir un tipo estructura `tConjuntoCartas` que defina listas de hasta 40 cartas y cuyas variables servirán para representar tanto el mazo como las cartas robadas por el jugador humano y por la máquina.

Recuerda que, en el modo D, cada partida debe comenzar con un mazo en memoria que contiene 40 cartas dispuestas de forma aleatoria. Una forma de conseguirlo es rellenar el mazo inicialmente con las 40 cartas disponibles y después barajarlas.

Además deberás incorporar, al menos, los siguientes subprogramas:

- `void inicializa (tConjuntoCartas & cartas)` Inicializa `cartas` a la lista vacía.
- `void sacarCarta (tConjuntoCartas & cartas, int & carta)` Elimina una carta de un extremo de la lista `cartas` y la devuelve en `carta`.
- `void annadirCarta (tConjuntoCartas & cartas, int carta)` Añade la carta `carta` en un extremo de la lista `cartas`.
- `void crearMazo (tConjuntoCartas & mazo)` Rellena `mazo` con 40 cartas dispuestas de forma aleatoria.
- `void modoDhumano (tConjuntoCartas & mazo, tCartasPorAparecer cartas, tConjuntoCartas & cartasHumano, float & puntos)` Permite realizar el turno del jugador humano en el modo D. Recibe el mazo y una variable `cartas` que indica cuántas cartas de cada tipo quedan en el mazo; al finalizar su ejecución deja `cartas` y `mazo` actualizados según las cartas que haya robado el humano en su turno y devuelve los puntos obtenidos por el humano así como las cartas que ha robado.
- `void modoDmaquina(tConjuntoCartas & mazo, tCartasPorAparecer cartas, float puntosHumano, tConjuntoCartas & cartasMaquina, float & puntos)` Permite realizar el turno del jugador máquina en el modo D. Recibe el mazo, una variable `cartas` que indica cuántas cartas de cada tipo quedan en el mazo y la puntuación obtenida por el jugador humano; al finalizar su ejecución deja `cartas` y `mazo` actualizados según las cartas que haya robado la máquina en su turno y devuelve los puntos obtenidos por la máquina así como las cartas que ha robado.

Además, incluye en tu programa dos constantes que nos permitan modificar el comportamiento del modo D simplemente cambiando sus respectivos valores antes de compilar. Una de las constantes indicará si queremos que se muestre en pantalla o no el contenido del mazo de cartas después de crearlo y justo después de hacer cualquier modificación en él. La otra constante indicará si, cuando llega el momento de crear y almacenar el mazo completo en memoria, deseamos crearlo de manera aleatoria (tal y como se ha indicado), o bien si preferimos cargarlo completo a memoria a partir de un fichero que nos indique el usuario (que seguirá el mismo formato que en los modos A, B y C). Estas constantes servirán para depurar el programa durante su desarrollo (fijaos en que el azar y la información oculta dificultan encontrar posibles errores) y, después de entregarlo, servirán para visualizar cómodamente su comportamiento interno durante la corrección con el profesor.

4. Versión 4 (opcional)

La versión opcional de la práctica consistirá en añadir un modo adicional de juego, el modo E. Dicho modo será similar al modo D, salvo en que supondremos que el jugador humano colocará cada nueva carta que robe en su turno *boca abajo* justo después de consultarla. Eso significa que, al llegar el turno de la máquina, ésta sabrá *cuántas* cartas ha sacado el humano antes de plantarse, pero no *cuánto suman* dichas cartas. Además, las cartas siempre se levantarán al finalizar la partida, por lo que la máquina podrá saber al terminar cada partida las cartas que recibió el humano, incluido en qué orden. Observando dicha información, la máquina podría intentar averiguar a lo largo de las partidas con qué puntuaciones suele plantarse el jugador humano y con cuáles no (o incluso, si suele mostrarse consistente en dicho criterio o no), y modificar consecuentemente su estrategia en sus partidas posteriores para tratar de aprovechar dicho conocimiento.

Separa dentro de tu fichero cpp todas las funciones que añadas sólo para la versión opcional, ponlas todas al final. No hagas el código propio de la versión opcional demasiado complejo, su tamaño no debería superar aproximadamente un tercio de lo que ocupe el código que hayas usado para la parte obligatoria de la práctica. Se valorará la capacidad de la máquina para ganar partidas al humano en función de su forma de jugar, así como la eficiencia del programa (no hagas complejos cálculos que ralenticen significativamente su ejecución).

5. Entrega de la práctica

La práctica se entregará en el Campus Virtual por medio de la tarea **Entrega de la Práctica 1**, que permitirá subir el archivo *main.cpp* con el código fuente de la versión 3 (o el archivo con la versión 4, si hiciste la parte opcional). Uno de los dos miembros del grupo será el encargado de subirlo, no lo suben los dos.

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