UNIVERSIDAD
COMPLUTENSE
MADRID

Fundamentos de la Programación

Presentación de la Práctica 1 (Versión 1)

(Basado en la práctica de Mercedes Gómez, Luis Hernández, Ramón González y Federico Peinado)

Índice

1. Introducción

- ✓ En esta práctica se pretende desarrollar, de manera incremental, distintas variantes del juego **las siete y media**.
- ✓ La práctica cubre los **temas 1 a 5** pasando por **tres versiones**.
- ✓ Algunos **objetivos** concretos son: bucles, subprogramas, entrada/salida, enumerados, arrays, structs sencillos y listas.
- ✓ La fecha de entrega es: **9 de diciembre**.
- ✓ La entrega debe realizarse a través del campus virtual, utilizando la *tarea de entrega de la práctica 1*. Debe subirse el archivo *main.cpp.* Sólo lo sube un miembro del grupo. Añadir comentarios con el nombre y apellidos de los dos integrantes del grupo.

2. Descripción del Juego

- ✓ En cada partida intervienen dos jugadores: el jugador humano y la máquina (en este orden).
- ✓ Usamos un mazo de **40 cartas** dispuestas aleatoriamente.
- ✓ Las 40 cartas se dividen en **cuatro grupos** (oros, copas, espadas y bastos) de diez cartas cada uno. Las cartas con un número asociado entre 1 y 7 se llaman como ese número. Las cartas que tienen asociado los números 10, 11 y 12 se llaman sota, caballo y rey, respectivamente.
- ✓ Cada carta tiene un **valor** asociado. Las cartas con número asociado entre 1 y 7 valen ese número. Si el número asociado es 10, 11 o 12 valen un medio.

- El **objetivo del juego** es que, en cada partida, cada jugador, a base de robar cartas del mazo de forma consecutiva, consiga acercarse a las siete y media, sin pasarse.
- La puntuación obtenida por un jugador en una partida es la suma de los valores de sus cartas.
- Si con las cartas robadas del mazo el jugador supera 7,5 puntos, pierde la partida y la gana su oponente.
- Si ninguno de los jugadores se pasa, gana el que más se acerque a 7,5 puntos.
- Si hay empate en el número de puntos, se elige ganador aleatoriamente.

2. Funcionalidad de la Aplicación (Versión 1)

- ✓ Cada versión avanza en objetivos y temario del curso.
- ✓ En esta versión, el mazo de 40 cartas está en un **fichero de texto**.
- ✓ El jugador puede elegir jugar en **modo A**, en **modo B** o **finalizar** el programa. El programa actúa **de forma cíclica**, permitiendo jugar hasta que se elige finalizar.
- ✓ Si el jugador elige jugar (A o B), el programa solicita el nombre del archivo donde se encuentran las cartas y se procede a jugar la partida. El menú vuelve a aparecer cuando ha terminado la partida.

- En el **modo A**, cada jugador está obligado a coger un **número** determinado de cartas que se genera aleatoriamente al comenzar la partida. Este número está en el rango [3, 5].
- En el **modo B** se genera aleatoriamente un número **max_cartas**, en el intervalo [3, 5]. Los jugadores no están obligados a coger max_cartas. Pueden plantarse antes.
- En este modo B, el jugador podrá coger cartas hasta llegar a max_cartas o hasta que se pase. Y la máquina se plantará cuando su puntuación supere la del humano.
- En ambos modos, para cada jugador se mostrará por pantalla cada carta robada y se mantendrá en memoria únicamente la puntuación que se consiga. Esta puntuación se irá mostrando actualizada tras cada robo de carta.
- Si el jugador o la máquina se pasan, finaliza la partida.

3. Detalles de Implementación (Versión 1)

- ✓ A continuación se dan algunas indicaciones para implementar la primera versión.
- ✓ Los **archivos de texto** con 40 cartas contienen 40 números. Cada número aparece en una línea. No hay información acerca de los palos.
- ✓ Para generar los **número aleatorios** hay que utilizar las funciones *rand()* y *srand()* de *cstdlib*. La secuencia comienza en un primer número que se denomina **semilla**. Para generar dicha semilla utilizar *srand()* con el argumento deseado. Usar, por ejemplo, *srand(time(NULL))* con la biblioteca *ctime*.
- ✓ Para generar número aleatorios en un rango usa %. Por ejemplo,

```
limiteInferior + rand() % (limiteSuperior+1-limiteInferior)
```

✓ Incluye 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).`