Práctica 2
Trucos de magia con cartas

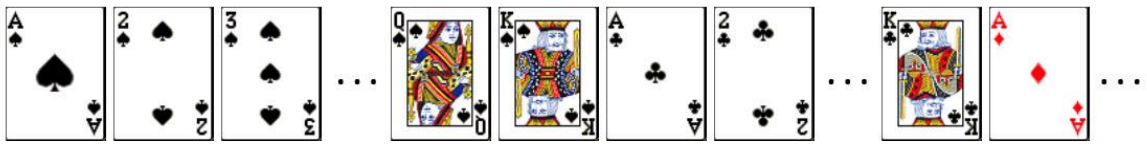
Fecha de entrega: 1 de febrero

1. Descripción de la práctica

El objetivo de esta práctica es programar un simulador de *trucos de cartas*. Trabajaremos con la baraja francesa, donde los palos son *picas*, *tréboles*, *diamantes* y *corazones*, y los "números" de cada palo son *A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q y K.*

Define los tipos enumerados `tPalo` y `tNumero`, con los valores indicados, y la constante `CARTASPORPALO` con el valor 13.

Las 52 cartas de la baraja están representadas por números enteros entre 0 y 51 (el 0 representará el *A (As) de picas*, el 1 el *2 de picas*, …, hasta el 51, que representará la *K de corazones*), dado que el orden de los palos es el que dijimos al principio:



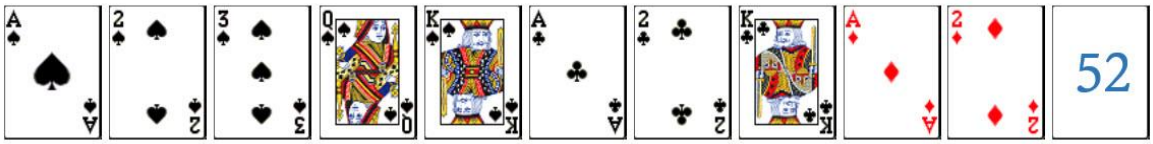
Define también el tipo `tCarta` como sinónimo de `int` y el tipo `tMazo` como un tipo de array de valores `tCarta` y tamaño `MAXCARTAS` (siendo `MAXCARTAS` 53). En los arrays de tipo `tMazo` marcaremos la primera posición no utilizada con un valor constante `CENTINELA` igual al entero "52" (como máximo se usarán 53 posiciones, de 0 a 51 para las 52 cartas posibles, más el centinela en la 53ª posición).

Versión 1: Manipulación de mazos de cartas

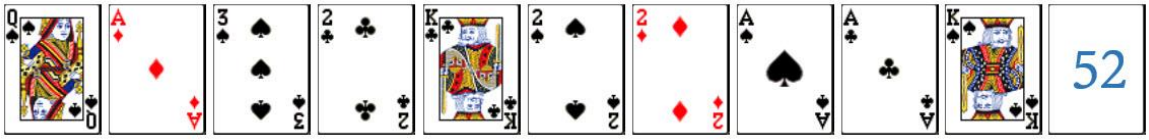
Al entrar en el programa, un menú mostrará al usuario las opciones disponibles para manipular un mazo de cartas (el programa no terminará hasta introducir *0*):

(1) **Cargar un mazo de cartas de fichero**. El programa pedirá al usuario que indique el fichero con el mazo a cargar. El programa pedirá repetidamente el nombre del fichero hasta que el fichero indicado pueda abrirse (o hasta un máximo de 3 veces). Si no se abre, mostrará un error y volverá al menú. En caso de poder cargar el mazo, mostrará por pantalla el mazo cargado.

(2) **Barajar mazo de cartas.** Barajará el mazo cargado actualmente en el programa. Por ejemplo, dado el siguiente mazo…

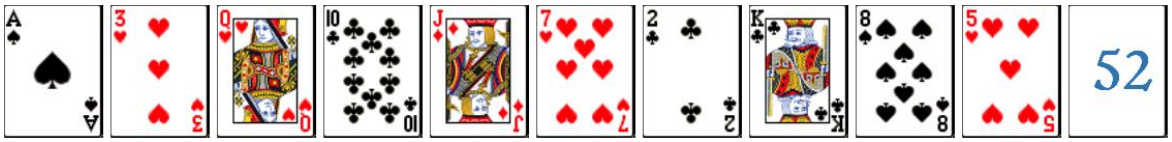


…un posible mazo resultante sería…

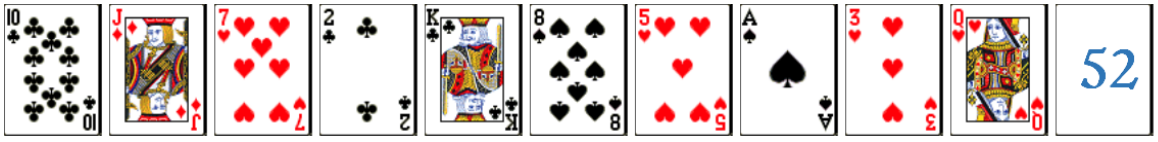


(3) **Añadir mazo**: El programa añadirá al final del mazo cargado en el programa un nuevo mazo que será leído desde fichero. Una vez concatenados los dos mazos mostrará el resultado por pantalla. En caso de superar el número máximo de cartas mostrará un error.

(4) **Cortar mazo:** Se pregunta al usuario por dónde se quiere dividir el mazo actualmente cargado en el programa. El mazo se corta por dicha posición, el primer montón resultante de dicha partición se pone tras el segundo montón restante y se muestra el mazo, ya todo junto. Por ejemplo, dado el mazo…



.. cortado por la *tercera carta* quedaría de la siguiente manera:



(5) **Guardar mazo de cartas en fichero:** Se pedirá al usuario que indique el nombre del fichero donde quiere guardar el mazo cargado actualmente en el programa y lo guardará en dicho fichero.

(0) **Salir:** Salimos del programa.

Para la implementación del programa utiliza *al menos* las siguientes funciones:

- ✓ **`void crearMazoVacio(tMazo mazo)`**: Hace que mazo pase a representar un mazo vacío colocando el centinela en la primera posición del array.
- ✓ **`int cuantasEnMazo(const tMazo mazo)`**: Devuelve cuántas cartas hay en el `mazo`.
- ✓ **`tPalo darPalo(tCarta carta)`**: Devuelve el palo de la carta. Por ejemplo, si le llega la carta de código 2 (correspondiente al *3 de picas*) devuelve *picas*.
- ✓ **`tNumero darNumero(tCarta carta)`**: Devuelve el "número" de la carta. Por ejemplo si le llega la de código 11 (*reina de picas*) devuelve *Q*.
- ✓ **`void escribirCarta(tCarta carta)`**: Muestra por pantalla una `carta`. Por ejemplo, si es la de código 2 (*3 de picas*) escribirá "3 de picas" en pantalla.
- ✓ **`void escribirMazo(const tMazo mazo)`**: Muestra por pantalla el contenido de mazo cargado actualmente en el programa.
- ✓ **`bool cargarMazo(tMazo mazo)`**: Si el archivo no puede abrirse, devuelve *false*. En otro caso, una vez abierto el archivo, leerá las cartas del fichero y las guardará en el `mazo`. Cada renglón del fichero contendrá una carta indicada con una letra (p, t, d, c) seguida de un número (entre 1 y 13). Por ejemplo, "c 4" será el *4 de corazones*, "t 10" será el *10 de tréboles* y "c 13" será el *rey de corazones*. El programa dejará de leer el fichero cuando encuentre el centinela "x" o se supere el máximo de cartas. Si se supera el máximo de cartas permitido se guardarán sólo las que entren en el mazo.
- ✓ **`void barajarMazo(tMazo mazo)`**: Baraja el `mazo`. Para ello se intercambian cartas dos a dos de forma aleatoria. Se harán (3 * numCartas) intercambios.
- ✓ **`bool unirMazos(tMazo mazoOriginal, const tMazo nuevoMazo)`**: Añade al final de `mazoOriginal` las cartas de `nuevoMazo` (en el mismo orden). Si la concatenación supera el máximo de cartas permitido en un mazo devuelve *false* (en otro caso *true*).
- ✓ **`bool partirMazo(tMazo mazoOrigen, int cuantasCoger, tMazo mazoDestino)`**: Quita a `mazoOrigen` sus primeras *`cuantasCoger`* cartas, formando con ellas, por orden, el mazo `mazoDestino`. ¡Fíjate en que esta función también modifica `mazoOrigen`! En caso de no haber suficientes cartas en `mazoOrigen` devuelve *`false`*, `mazoOrigen` no debe verse modificado y `mazoDestino` se devuelve vacío.

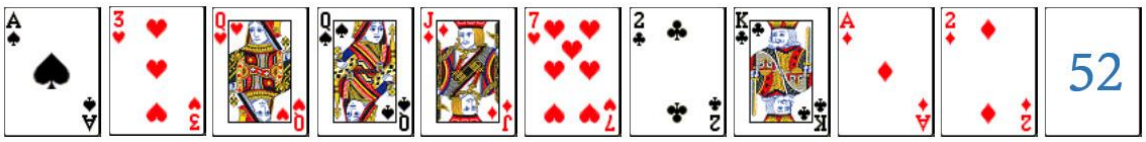
- ✓ **`void cortarMazo(tMazo mazo, int cuantas);`** Corta `mazo` por *`cuantas`*. Para ello, se parte en un mazo auxiliar (el primer montón), que después será concatenado al resto (segundo montón). Si *`cuantas`* no es válido (porque no hay tantas cartas), entonces se entenderá que no se quiere modificar `mazo`, y se acabó. Esta función utilizará la función anterior para partir el mazo.
- ✓ **`void guardarMazo(const tMazo mazo);`** Pide al usuario el nombre del fichero en el que guardar el `mazo` y lo escribe en dicho fichero, siguiendo exactamente *el mismo formato* que usamos en los ficheros de lectura de mazos.
- ✓ **`int menu();`** Muestra al usuario el menú de opciones y le pide una opción. Repite la petición al usuario mientras la opción no sea válida. Cuando obtiene una opción válida, la devuelve.

Asegúrate de que tus funciones utilizan las otras funciones que ofrecen lo que necesitas, ¡no vuelvas a programarte algo que ya te da hecho otra función! ☺

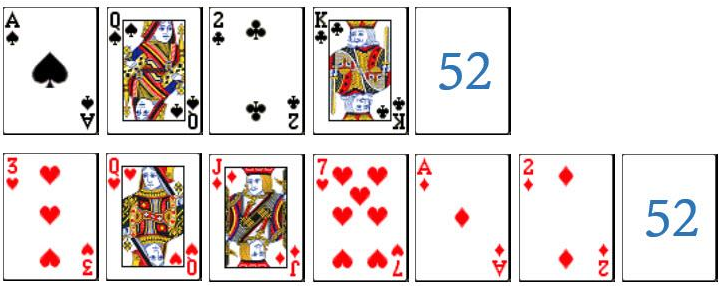
Versión 2: Repartos de mazos de cartas

Añade las siguientes opciones para dividir el mazo del programa según distintos criterios:

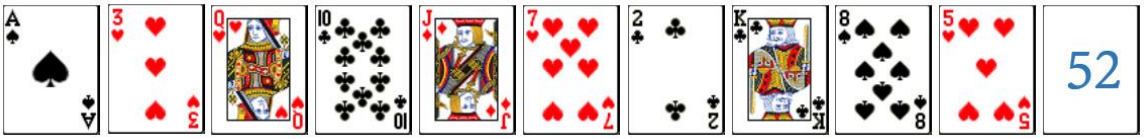
(6) **Separar en negras y rojas:** Separa el mazo del programa en cartas negras (picas y tréboles) y cartas rojas (diamantes y corazones). Por ejemplo, dado el mazo…



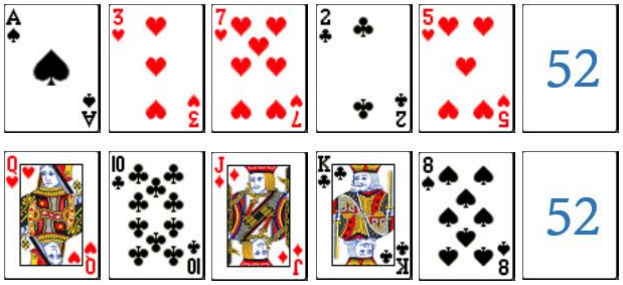
…los dos mazos resultantes serían…



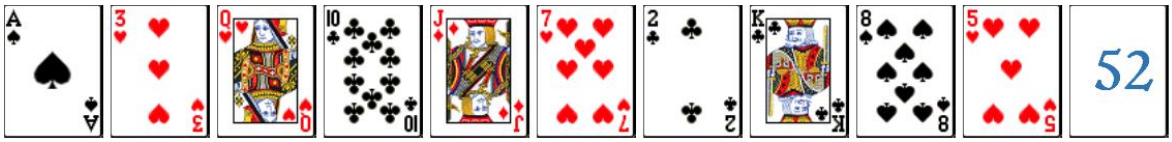
(7) **Separar en bajas y altas**: Separa el mazo del programa entre cartas bajas (A-7) y cartas altas (8-K). Por ejemplo, dado el mazo…



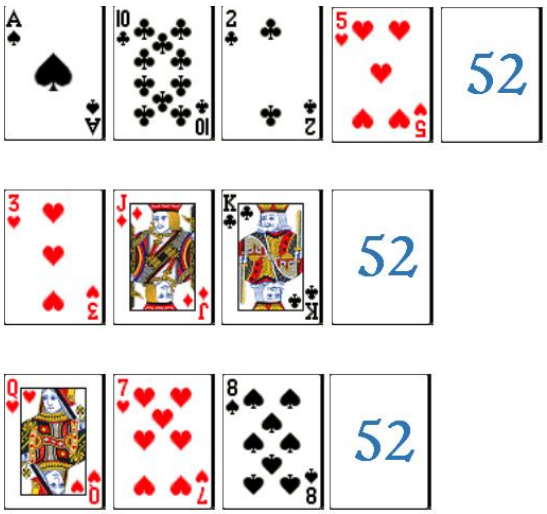
…los dos mazos resultantes serían…



(8) **Repartir un mazo de manera alterna en 3 montones**: Repartimos alternativamente (repartiendo carta por carta) el mazo del programa en 3 montones. Por ejemplo, dado el mazo…



…los tres mazos resultantes serían…



Para la implementación del programa utiliza *al menos* las siguientes funciones:

- ✓ **`void repartirNegroRojo(const tMazo mazo, tMazo mazoNegro, tMazo mazoRojo)`:** Separa el `mazo` en dos mazos, uno que contenga las cartas negras (picas y tréboles) y otro que contenga las cartas rojas (diamantes y corazones).
- ✓ **`void repartirBajaAlta(const tMazo mazo, tMazo mazoBajas, tMazo mazoAltas)`:** Separa el `mazo` en dos mazos, uno que contenga las cartas bajas (desde A hasta 7) y otro que contenga las cartas altas (desde 8 hasta K).
- ✓ **`void repartirIntercalando(const tMazo mazo, int enCuantosMazos, int queMazoPido, tMazo mazoNuevo)`:** `mazoNuevo` pasa a contener el *`queMazoPido`*-ésimo mazo que se obtendría al repartir `mazo` alternamente en *`enCuantosMazos`* mazos. Por ejemplo, si el mazo contuviera las cartas 1, 2, 3, 4, 5, 6 y 7, entonces `repartirIntercalando(mazo, 3, 1, mazoNuevo)` haría que `mazoNuevo` se convirtiera en un mazo con 1, 4 y 7. Por otro lado, `repartirIntercalando(mazo, 3, 2, mazoNuevo)` haría que mazoNuevo contuviera las cartas 2 y 5.

Asegúrate de que tus funciones utilizan las otras funciones que ofrecen lo que necesitas, ¡no vuelvas a programarte algo que ya te da hecho otra función! ☺

Versión 3: Truco de los tres montones

Añade al menú una opción para ejecutar el llamado "Truco de los tres montones". El truco de los tres montones consiste en lo siguiente:

- El mago baraja un mazo con 21 cartas y las reparte en tres mazos, pidiendo a un espectador que memorice una de las 21 cartas que ha repartido sin decir cuál es, pero que diga en cuál de los tres mazos ha caído dicha carta.
- Entonces el mago junta los tres mazos en uno solo, concatenando los tres y poniendo el mazo que ha indicado el espectador en medio de los otros dos.
- Después el mago reparte el nuevo mazo en tres mazos repartiendo las cartas de manera alterna (mazo 1, mazo 2, mazo 3, mazo 1, mazo 2...) y vuelve a pedir al espectador que diga en qué mazo está ahora la carta que escogió antes.
- El mago vuelve a concatenar los tres montones, poniendo el mazo indicado por el espectador en medio.
- Entonces el mago vuelve a hacer lo mismo con el espectador por tercera vez.
- Ahora, la carta elegida por el espectador es la que está en la posición 11 del mazo, así que el mago se la enseña sin dudas al espectador. ¡Tachán! ☺

¿Qué debe hacer el programa *exactamente* para realizar este truco?

- El programa pedirá al usuario el nombre del fichero del que se leerá el mazo de partida (el truco funcionará con un mazo de 21 cartas diferentes; con otros mazos habría otros efectos).
- Tras leer el mazo, el programa lo barajará y repartirá el mazo resultante en tres montones de manera alterna. Entonces el contenido de los tres mazos se mostrará al usuario.
- Se pedirá al usuario que escoja mentalmente una carta y que diga en qué mazo (1, 2 ó 3) está su carta elegida.
- Los tres mazos se unirán entonces en uno solo, poniendo el mazo indicado por el usuario en medio de los otros dos.
- Entonces el proceso de repartir las cartas en tres mazos, preguntar al usuario en qué mazo está su carta y juntar los tres mazos se volverá a repetir otras dos veces, aunque ahora además se recordará al usuario que la carta debe ser la misma que antes.
- Después el programa indicará al usuario la carta que eligió (la undécima del mazo final).

Para la implementación del truco añade *al menos* la siguiente función:

- ✓ **`void trucoTresMontones():`** Lleva a cabo el truco de los tres montones descrito anteriormente. Esta función debe usar las funciones ya hechas en las versiones anteriores (cuando proceda).

Versión 4: Truco de la posada

Añade al menú de opciones el llamado "Truco de la posada". El truco de la posada consiste en lo siguiente:

- El mago cuenta que había una posada con cuatro habitaciones. Llegaron cuatro caballeros (las cuatro J) y cada uno se puso en una habitación diferente (cada carta se pone en un sitio de la mesa). Luego llegaron cuatro señoras (Q) y, para no dejarlas sin habitación, ubicaron a cada una en una de dichas habitaciones, con los caballeros (cada Q se pone sobre cada J). Luego llegaron cuatro reyes (K) con sus cuatro peones (A), y pusieron cada rey y cada peón en alguna de dichas cuatro habitaciones. Así que en cada habitación quedó una J, una Q, una K y una A (es decir, tenemos cuatro mazos, cada uno con una J, una Q, una K y una A). Entonces se fueron todos a dormir.
- El mago recoge los cuatro mazos, concatenándolos, y pide a un espectador que corte el mazo por donde quiera.

- Después el mago reparte las cartas del mazo resultante de manera alterna en cuatro mazos (mazo 1, 2, 3, 4, 1, 2,...), que vuelven a representar las cuatro habitaciones. Estos muestran que, a la mañana siguiente, los cuatro reyes amanecieron juntos en la misma habitación (es decir, en el mismo mazo), las cuatro señoras en otra (en otro mazo), los cuatro caballeros en otra, y los cuatro peones en otra. ¡Tachán! ☺

¿Qué debe hacer el programa *exactamente* para realizar este truco?

- El programa pedirá al usuario el nombre del fichero que contenga el mazo a usar (el truco funcionará como se ha dicho arriba si el fichero contiene primero las cuatro J, luego las cuatro Q, luego las cuatro K y luego los cuatro A; otros mazos tendrán otros efectos).
- Tras leer el mazo del fichero, se repartirá alternamente en cuatro mazos y se mostrará en pantalla el contenido de los cuatro mazos.
- Entonces los cuatro mazos se concatenarán en uno solo y se pedirá al usuario que indique en qué número de carta quiere cortar dicho nuevo mazo.
- El mazo se cortará por el punto indicado (es decir, se separará en dos mazos primero y segundo, y luego ambos mazos se concatenarán poniendo el primero después del segundo), se repartirá en cuatro mazos de manera alterna como antes, y se mostrarán dichos mazos al usuario.

Para la implementación del truco añade *al menos* la siguiente función:

- ✓ **`void trucoPosada():`** Lleva a cabo el truco de la posada descrito anteriormente. Esta función debe usar las funciones ya hechas en las versiones anteriores (cuando proceda).

Parte opcional: El truco del jugador desconfiado

Añade al menú una nueva opción para el llamado "Truco del jugador desconfiado". El truco del jugador desconfiado consiste en lo siguiente:

- Un jugador de póker, llamado Jugador 1, dice que desconfía de que los demás jugadores sentados con él en la mesa (llamados Jugador 2, Jugador 3 y Jugador 4, yendo en sentido contrario a las agujas del reloj desde Jugador 1) le hagan trampas. Por ello, propone que las 20 cartas que se van a repartir sean visibles para todos antes de repartirse (A, Q, K de picas; 2, 4, 5, 6, 9, Q, K de tréboles; 2, 3, 4, 5, 6, 10, K de diamantes; y 10, Q, K de corazones).
- Ese mazo de 20 cartas se baraja y se reparte entre los cuatro jugadores (5 cartas para cada uno).
- Jugador 1 no se fía del jugador que ha repartido, así que propone seguir unos pasos para mezclar ese reparto y asegurarse de que realmente salga un reparto

aleatorio y justo. Cada jugador dividirá privadamente sus cartas en dos mazos, sin mostrar nunca sus cartas, de la siguiente manera:

- Jugador 1 pone sus cartas negras a su izquierda y sus cartas rojas a su derecha.
- Jugador 2 pone sus cartas desde A hasta 7 a su izquierda y desde 8 hasta K a su derecha.
- Jugador 3 pone sus cartas pares (incluyendo Q=12) a su izquierda e impares (incluyendo J=11 y K=13) a su derecha.
- Jugador 4 pone sus cartas figura o as (A, J, Q, K) a su izquierda y el resto (números) a su derecha.

- Entonces cada jugador da al jugador a su izquierda su mazo izquierdo y a su jugador a su derecha su mazo derecho, todos los jugadores a la vez.
- Con las nuevas cartas recibidas por cada uno de esa manera, los cuatro vuelven a hacer la misma división en dos mazos usando el mismo criterio que antes, y vuelven pasarlas a sus compañeros de la misma forma.
- Entonces vuelven a hacer esa división con las cartas recibidas una tercera vez, pero esta vez se quedan para ellos su mazo derecho, y simplemente pasan a su vecino izquierdo su mazo izquierdo.
- Ahora Jugador 1 dice que, por fin, el reparto es realmente caótico y aleatorio, así que se puede fiar de él y ya pueden jugar la partida con esas cartas. Pero, sorpresa, resulta que ahora las cartas de cada jugador son las siguientes:
  - Jugador 1 tiene escalera de color.
  - Jugador 2 tiene full.
  - Jugador 3 tiene póker.
  - Jugador 4 tiene color.

  Así que Jugador 1 ve cómo sus contrincantes apuestan fuerte en esta mano, pues todos creen tener jugadas ganadoras. Pero quien gana dicha gran apuesta es siempre Jugador 1, ya que su jugada es la mejor. ¡Tachán! ☺

¿Qué debe hacer el programa *exactamente* para realizar este truco?

- El programa pedirá al usuario el fichero del que tomar el mazo (el truco saldrá como está previsto con el mazo indicado arriba; con otros mazos los efectos serán diferentes).
- El mazo así cargado se barajará, después se mostrará en pantalla, y luego las cartas del mazo se repartirán alternamente en los cuatro mazos que representarán las cartas de cada jugador.

- Las cartas recibidas por cada jugador se mostrarán en pantalla (en el truco, las cartas de cada jugador no se muestran nunca, pero el programa sí lo hará para que veamos que todo va bien).
- Entonces el mazo de cada jugador se dividirá en dos mazos, izquierdo y derecho, tal y como se indica arriba, aplicando a cada mazo el criterio de su jugador correspondiente.
- Se usarán los ocho mazos resultantes del paso anterior para crear cuatro mazos de la manera siguiente: para cada jugador, crearemos un nuevo mazo juntando el mazo izquierdo de su vecino derecho con el mazo derecho de su vecino izquierdo.
- El programa mostrará cómo quedarán las cartas de los cuatro jugadores tras el proceso anterior.
- El proceso de dividir los mazos de todos los jugadores en dos mazos, usar los ocho mazos resultantes para crear los nuevos cuatro mazos de los jugadores, y mostrar los cuatro mazos resultantes, se repetirá otra vez.
- Entonces los jugadores volverán a crear sus mazos izquierdo y derecho, pero esta vez nuevo el mazo de cada jugador se creará juntando su propio mazo derecho con el mazo derecho de su vecino izquierdo.
- Finalmente, se mostrarán las cartas de los cuatro jugadores.

Para la implementación del truco añade *al menos* las siguientes funciones:

- ✓ **`void repartirParImpar(const tMazo mazo, tMazo izq, tMazo der):`** Hace que *`izq`* contenga las cartas pares de `mazo` (incluida Q) y *`der`* las cartas impares de `mazo` (incluidas J y K).
- ✓ **`void repartirFiguraNoFigura(const tMazo mazo, tMazo izq, tMazo der):`** Hace que *`izq`* contenga las cartas figuras o as de `mazo` (A, J, Q, K) y *`der`* las demás cartas de `mazo` (2-10).
- ✓ **`void trucoJugadorDesconfiado():`** Lleva a cabo el truco del jugador desconfiado.

Curiosidades de la práctica ¿Entiendes por qué funcionan los tres trucos? En el truco del jugador desconfiado, ¿puedes inventar otros mazos con los que puedas asegurar que cada jugador acabará con unas cartas determinadas? Dicho tercer truco ha sido inventado por un profesor de la asignatura, Ismael Rodríguez, pero nos hemos tenido que inventar el nombre de los otros dos trucos porque no los sabemos (si tienen nombre y lo conoces, ¡dínoslos, por favor! ☺).

2. Requisitos de implementación

No olvides declarar las constantes necesarias para hacer tu código reutilizable y fácil de modificar.

No olvides incluir los prototipos de tus funciones.

No utilices variables globales: cada función, además de los parámetros con los que se le pasa información en las llamadas, debe declarar localmente las variables que requiera.

No pueden usarse en la resolución de la práctica elementos de C++ no vistos en clase. El programa no debería utilizar instrucciones de salto no estructuradas como exit, break (salvo en las cláusulas de la instrucción switch) y return (salvo en la última instrucción de las funciones que devuelven un valor).

3. Entrega de la práctica

La práctica se entregará antes de la fecha de entrega en el espacio online de la asignatura dentro del Campus Virtual. La entrega se realizará a través de la tarea **Entrega Práctica 2**, que permitirá subir, como mínimo, un archivo con el código fuente de la última versión que llegues a implementar de la práctica (`practica2.cpp`). Asegúrate de poner todos los datos de la asignatura y de los miembros del grupo en los comentarios que van al inicio de los ficheros de código fuente.

El fichero subido deberá tener el siguiente nombre: P2GXX.zip, siendo XX el número de vuestro grupo. Uno sólo de los miembros del grupo (no los dos) será el encargado de subir la práctica.

La práctica debe funcionar usando Visual Studio, que es la herramienta vista en clase, en la versión que tenemos disponible en el laboratorio. La práctica subida debe, por supuesto, compilar y cumplir con exactitud con la funcionalidad descrita en este enunciado.