PROYECTO INFORMÁTICA DOMINÓ

El programa empieza con un menú de juego, el cuál te permitirá escoger entre diferentes opciones:

- **Modo solo**: Habrá un modo solitario, ya explicaré más adelante en que consiste.
- **Modo multijugador**: Permite jugar con varios jugadores, siendo el jugador 1 controlado por el usuario y el resto por el ordenador. Dentro de esta modalidad, te permitirá escoger:
  - El número de jugadores (entre 2 y 4)
  - La dificultad del juego. Explicaré más adelante la diferencia entre modo fácil y difícil.

Pasemos a describir todos los modos de juego.

1. MODO MULTIJUGADOR

El sistema principal de juego viene incluido en una función denominada modomultijugador, la cuál recurre a diversas subfunciones a lo largo del programa destinadas a llevar a cabo diversos trabajos durante la partida. Esta función es un void que necesita dos parámetros, el número de jugadores y la dificultad, ya veremos por qué.

1.1. CREACIÓN DE LAS FICHAS

El primer paso es crear las fichas de juego. Para ello voy a emplear una estructura (ficha) que consta de 2 números int. Para almacenarlas, voy a crear un vector que contenga a todas las fichas, de forma que el ordenador asociará a cada una la posición que ocupa en él. Para simbolizar no haber ficha, voy a usar el 0, razón por la cuál la "ficha" de la primera posición del vector "fichas" no existe, pues no voy a usarla nunca.

Como en la realidad puedo colocar la ficha de dos formas, es decir, [1,2] o [2,1], usaré números negativos, de forma que, si el número asociado a la ficha es negativo, le dará la vuelta a la misma.

1.2. REPARTO DE LAS FICHAS

Lo siguiente que debemos hacer es repartir las fichas. Para ello, voy a seguir 2 pasos:

- Primero voy a rellenar un vector ("valores") con números aleatorios entre 1 y 28 sin repetición (dado que no puede haber dos fichas iguales). Para ello, empleo un bucle do-while, de forma que, si el número generado coincide con alguno de los anteriores, se volverá a generar oro número sin introducirlo en el vector.
  Con ese objetivo, está la variable igual, que detecta si algún número se repite.

```
srand(time(NULL));
for(i=0; i<28; i++)
{
    do
    {
        igual=0;
        aleatorio=1+(rand()%28);
        for(j=0; j<i; j++)
        {
            if(aleatorio==valores[j])
            igual=1;
        }
    }while(igual==1);
    valores[i]=aleatorio;
}
```

Obsérvese que los números aleatorios son siempre positivos

- Después, voy a asociar a cada jugador 7 fichas. No obstante, la forma de repartir será diferente según el número de jugadores seleccionado:
  - **2 jugadores**: Asociará 7 fichas consecutivas al primer jugador, las 7 siguientes al segundo y el resto irán al pozo ("pozo" sirve para acumular las fichas restantes de forma que si un jugador no puede colocar, tenga que robar obligatoriamente).
  - **3 jugadores**: Asociará 7 fichas consecutivas al primer jugador, 7 al segundo, 7 al tercero y el resto irán al pozo.
  - **4 jugadores**: Asociará 7 fichas a cada jugador. Por tanto, no hay pozo y nadie puede robar ficha en caso necesario.

Obsérvese que he creado un vector distinto para almacenar las fichas de cada jugador

Algo importante, es la variable contadorpozo, que sirve para conocer en todo momento la última posición del vector "pozo" ocupada. Por tanto, si la variable vale -1, significa que no hay más fichas disponibles para robar.

1.3. PRIMER TURNO DE JUEGO

Este turno se realizará siempre automáticamente, pues todas las partidas empiezan de la misma forma. El jugador que comienza la partida será el que contenga la ficha más alta. Para ello, debemos conocer dos cosas:

- **¿Cuál es la ficha más alta en juego?:** Para ello, vamos a emplear una función, denominada "fichaalta". Lo que hace esta función es recorrer el vector de las fichas de cada jugador y ver cuál es el número más alto. Esta función devuelve un valor, el asociado a la ficha más alta. No obstante, si el número de jugadores es 4, este número siempre será el 28, es decir, el [6,6].

Obsérvese que, al crear las fichas, he asociado los números más altos a las fichas dobles, pues a la hora de empezar tienen prioridad

- **¿Quién tiene esa ficha?:** Para ello, vamos a emplear una función, denominada "primerturno". Esta función va a recorrer el vector de las fichas de cada jugador para ver quien tiene la ficha más alta. Va a devolver un valor diferente dependiendo de quien tenga la ficha:
  - **Jugador 1:** Devuelve un 0.
  - **Jugador 2:** Devuelve un 1.
  - **Jugador 3:** Devuelve un 2.
  - **Jugador 4:** Devuelve un 3.

¿Por qué he escogido esos números?

Para entenderlo, voy a explicar la función "siguienteturno". La sucesión de turnos es la siguiente:

- Si hay dos jugadores: 0, 1, 0, 1, 0, 1…
- Si hay tres jugadores: 0, 1, 2, 0, 1, 2…
- Si hay cuatro jugadores: 0, 1, 2, 3, 0, 1, 2, 3…

Si nos damos cuenta, en los tres casos, los números de la sucesión coinciden con los posibles restos de dividir cualquier número entre el número de jugadores.

```
int siguienteturno(int jugadores, int turno)
{
    if(jugadores==2)
        return (turno+1)%2;
    if(jugadores==3)
        return (turno+1)%3;
    if(jugadores==4)
        return (turno+1)%4;
}
```

Una vez determinado quién empieza el turno y qué ficha debe poner, podemos ejecutar el inicio de la partida. Previamente, voy a mostrar en pantalla las fichas repartidas a cada jugador y las restantes en el pozo. Para mostrar en pantalla el tablero de juego, voy a emplear la función "juego", la cuál me imprime, tras cada turno, las fichas de cada jugador, las puestas en el tablero, las restantes en el pozo y me indica de quién es el turno.

```
Tablero:

Jugador 1: [1|2][2|5][6|6][3|3][2|3][0|3][3|6]
Jugador 2: [0|1][0|5][3|4][5|5][5|6][1|4][1|1]
Jugador 3: [2|4][2|2][1|3][0|4][1|5][1|6][4|6]
Jugador 4: [0|2][0|0][2|6][4|5][0|6][4|4][3|5]

Pozo:

                TURNO DEL JUGADOR 1
```

4 jugadores

```
Tablero:

Jugador 1: [0|3][2|5][0|6][0|2][5|6][3|5][0|5]
Jugador 2: [3|6][1|4][1|6][6|6][4|4][3|3][0|1]
Jugador 3: [2|6][1|5][2|2][1|2][4|5][1|3][5|5]

Pozo:   [0|4][3|4][1|1][4|6][2|4][2|3][0|0]

                TURNO DEL JUGADOR 2
```

3 jugadores

```
Tablero:

Jugador 1: [4|6][2|2][3|5][3|4][4|4][0|0][3|3]
Jugador 2: [4|5][1|3][0|1][5|5][0|5][6|6][5|6]

Pozo:   [1|4][0|6][3|6][1|6][2|6][0|3][0|2][1|5][1|2][2|4][2|5][2|3][0|4][1|1]

                TURNO DEL JUGADOR 2
```

2 jugadores

Para llevar a cabo el primer turno, introduce la ficha más alta en el tablero, se la quita al jugador correspondiente y avanza 1 turno. Posteriormente, vuelve a mostrar por pantalla el estado actual de la partida.

1.4. DESARROLLO DEL JUEGO

A partir de aquí, se permitirá al usuario el control del jugador 1. El desarrollo de la partida dependerá de la dificultad de juego escogida (no obstante, la forma de controlar al jugador local será la misma en ambos casos), por lo que podemos diferenciar entre:

1.4.1. MODO FÁCIL

En este modo el ordenador no podrá pensar que ficha la conviene jugar. Lo que va a hacer es recorrer una a una las fichas del jugador (dependiendo del turno, empleará un vector de fichas u otro), de forma que colocará la primera que pueda poner. En caso negativo, robará una ficha del pozo (si quedan) y volverá a ser su turno, de forma que el turno no avanza hasta que un jugador pone ficha o, a pesar de no poder poner, no quedan fichas en el pozo.

¿Cómo sabe el ordenador si puede poner o no una ficha?

De todos los números de las fichas del tablero, solo nos importan 2, el primer número de la primera ficha y el segundo de la última ficha. Por tanto, tendrá que ver si algún número de la ficha del jugador coincide con alguno de esos 2.

A la hora de repartir las fichas, los números asociados a todas ellas son positivos, por lo que, en los vectores de las fichas de los jugadores, estas aparecerán tal y como están definidas en la estructura. Sin embargo, el número asociado a las fichas del tablero no tiene porqué ser positivo, pues puedo haber necesitado dar la vuelta a la ficha para poder colocarla. Por ello, tenemos 4 opciones:

- **tablero [0]>0 y tablero[contadortablero]>0** (la variable contadortablero determina cuál es la última posición del vector "tablero" ocupada)

Podemos resumir todas las fichas del tablero como una sola con los siguientes números:

$fichas[tablero[0]].numero1$	$fichas[tablero[contadortablero]].numero2$

La ficha del jugador, que siempre será la misma, es:

$fichas[fichasjugador\lambda[i]].numero1$	$fichas[fichasjugador\lambda[i]].numero2$

- **tablero[0]>0 y tablero[contadortablero]<0**

$fichas[tablero[0]].numero1$	$fichas[tablero[contadortablero]*(-1)].numero1$

$fichas[fichasjugador\lambda[i]].numero1$	$fichas[fichasjugador\lambda[i]].numero2$

- **tablero[0]<0 y tablero[contadortablero]>0**

$fichas[tablero[0]*(-1)].numero2$	$fichas[tablero[contadortablero]].numero2$

$fichas[fichasjugador\lambda[i]].numero1$	$fichas[fichasjugador\lambda[i]].numero2$

- **tablero[0]<0 y tablero[contadortablero]<0**

$fichas[tablero[0]*(-1)].numero2$	$fichas[tablero[contadortablero]*(-1)].numero1$

$fichas[fichasjugador\lambda[i]].numero1$	$fichas[fichasjugador\lambda[i]].numero2$

A la hora de ver si puedo colocar, primero miro el signo del número del tablero, en la primera y última posición; y luego veo si los números de las fichas coinciden. En caso de poder colocar:

- Sumaré uno a la variable contadortablero (pues hay una ficha más en él)
- Asociaré la variable "ganador" al jugador que haya colocado (de esta forma, gana el último jugador en colocar)
- La variable "final" valdrá 0, ya veremos más adelante por qué.
- La variable "repetición" vale 1. Esto sirve para 2 cosas:
    - Que cada jugador solo pueda poner 1 ficha por turno, pues una condición para poner ficha es que repetición valga 0).
    - Detectar si debe o no robar ficha.

¿Qué ocurre en caso de no poder poner?

Si después de analizar todos los casos posibles el jugador no ha puesto ficha (eso se detecta mediante la variable "repetición", de forma que si no ha colocado vale 0), tendrá que robar una ficha del pozo (usando la función "robarficha" y solo cuando queden fichas en él).

```
void robarficha(int jugador[], int pozo[], int contadorpozo)
{
    int i;
    for(i=0; i<27; i++)
        jugador[27-i]=jugador[26-i];
    jugador[0]=pozo[contadorpozo];
    pozo[contadorpozo]=0;
}
```

En caso de robar ficha:

- Resto 1 a la variable contadorpozo (pues hay una ficha menos en él)
- Imprimo el tablero de juego, de esta forma puedo ver cada vez que un jugador robe ficha.

¿Cuándo acaba la partida?

En el dominó hay dos formas de acabar la partida:

- **Un jugador se queda sin fichas**: Tras cada turno, voy a comprobar si hay algún jugador que no tenga fichas. Para ello, voy a emplear la función "comprueba", que mira si, en el vector de las fichas del jugador que acaba de tener turno, todos los números son 0 (es decir, no tiene fichas), la partida termina.
- **Nadie puede poner ficha**: Otra forma de acabar la partida es la siguiente. Si un jugador coloca y, al volverle a tocar a él, ningún otro jugador ha podido colocar, ese jugador gana automáticamente. Esa es la razón por la cuál cuando un jugador coloca ficha la variable "final" vuelve a 0 y, si no puede colocar, le sumo 1.

Una vez se acaba la partida, te muestra por pantalla quién ha sido el ganador.

```
int comprueba
{
    int i, sinfichas=1;
    for(i=0; i<28; i++)
    {
        switch(turno)
        {
            case 0:
                if(fichasjugador1[i]!=0)
                sinfichas=0;
                break;
            case 1:
                if(fichasjugador2[i]!=0)
                sinfichas=0;
                break;
            case 2:
                if(fichasjugador3[i]!=0)
                sinfichas=0;
                break;
            case 3:
                if(fichasjugador4[i]!=0)
                sinfichas=0;
                break;
        }
    }
    if(sinfichas==1)
        final=4;
    return final;
}
```

1.4.2. MODO DIFÍCIL

Lo que voy a intentar es que el ordenador pueda decidir que ficha la conviene poner, con el objetivo de que el resto de jugadores no puedan hacerlo. Para ello, sigo una serie de pasos:

- Cuento cuantas veces aparece cada número entre todas las fichas presentes en el tablero.
- Ordeno de mayor a menor el número de repeticiones de cada número.
- Cambio el número de repeticiones por el número repetido

A partir de aquí, el procedimiento para poner ficha es el mismo que en el modo fácil, con una condición más:

El número que acompaña al que coincide con el del tablero debe ser el que más veces de repite. En el caso de que no se pueda, pasa al segundo que más veces se repite y así hasta el último

1.4.3. JUGADOR LOCAL