UNIVERSIDAD COMPLUTENSE MADRID

Fundamentos de la Programación I

Presentación de la Práctica (Versión 2)

(Basado en la práctica de Luis Garmendia, Ramón González del Campo, Pablo Rabanal y Luis Hernández)

Índice

1. Funcionalidad de la Aplicación

- ✓ En esta segunda versión vamos a utilizar **arrays** para almacenar las fichas que se pueden robar en la mesa (llamada pozo, inicialmente con las 28 fichas posibles) y las fichas del jugador.
- ✓ **Inicio del Juego**
  - Se debe generar el pozo con las 28 fichas posibles.
  - Se roba al azar una ficha del pozo y se coloca en el tablero.
  - El jugador roba al azar 7 fichas del pozo.
  - Las fichas recogidas del pozo se eliminan del pozo.

✓ Desarrollo del Juego

- Antes de cada jugada se muestran las fichas colocadas en la mesa (tablero), el número de fichas colocadas, el número de fichas robadas y las fichas restantes.
- Las **opciones** que puede elegir el jugador 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.

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

    Si se coloca la ficha 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.
  En este caso se pregunta cuál de las fichas del jugador se desea colocar a la derecha del tablero.

  Sólo se puede colocar una ficha si uno de sus valores coincide con el valor del extremo del tablero.
  Si se coloca la ficha debe representarse en el tablero en la posición correcta.
  La ficha colocada se elimina de la lista de fichas del jugador.

- **3) Robar** una ficha aleatoriamente del pozo que se elimina del pozo y se añade a la lista de fichas del jugador.

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

- **4) Salir** del juego. En este caso se muestran los puntos del jugador y se termina el programa.

✓ Final del Juego

- Una ronda de dominó termina en uno de los dos siguientes casos:
  - **1)** Si el jugador coloca todas sus fichas.
  - **2)** Si no quedan fichas para robar en el pozo y el jugador no puede poner ninguna ficha. En este caso se debe mostrar los puntos del jugador antes de salir.
  - **3)** Si se selecciona la opción de Salir

FP1: Presentación de la Práctica (Versión 1)

```
 ------------------
|      TABLERO     |
 ------------------
|1-1||1-4||4-5||5-6||6-6|

Fichas colocadas: 4 - Fichas robadas: 1
Fichas del jugador: |0-2| |2-5| |2-3| |1-2|

 ------------------
| MENU DE OPCIONES |
 ------------------
1. Poner ficha por la izquierda
2. Poner ficha por la derecha
3. Robar ficha nueva
0. Salir

Elija opcion: 1
```

✓ Funcionalidades Opcionales

- Salvar la partida en un archivo, para cargarla más adelante y continuar con ella.
- Permitir que el programa se pueda configurar para jugar con otras variantes del juego: entre 6 y 9 como máximo de puntos de las fichas.

2. Detalles de Implementación

- ✓ En la versión 2 el pozo y las fichas del jugador se almacenan en dos **arrays** de enteros que representan el valor izquierdo y derecho de cada ficha, junto con una variable entera contador de fichas.

```
const int NumFichas = 28;
typedef short int tArray[NumFichas];
```

- ✓ El **tablero** se puede seguir almacenando en un string.
- ✓ Además deberás incorporar los **subprogramas** necesarios decidiendo el tipo de retorno, el tipo de cada argumento y si cada argumento se pasa por valor o por referencia.

- ✓ Algunas de dichas funciones podrían ser las siguientes:

  - `void generaPozo(tArray pozo1, tArray pozo2, int maxValor); //Genera en pozo1 y pozo2 las partes izquierda y derecha de todas las piezas posibles del dominó para fichas con valores de 0 a maxValor`
  - `void mostrarTablero(string tablero, short int numColocadas, short int numRobadas, const tArray fichas1, const tArray fichas2, short int fichasCont); //Escribe en pantalla toda la información visible de la situación actual de la partida, incluyendo las fichas del jugador codificadas en fichas1, fichas2 y fichasCont (a esta función no le concierne el pozo, pues no se muestra)`
  - `void robaFicha(const tArray pozo1, const tArray pozo2, short int &cont, short int &fichaN1, short int &fichaN2); //Quita la última ficha del pozo (codificado en pozo1, poz2 y cont) y pone dicha ficha en fichaN1 y fichaN2`

- `void eliminaFicha (tArray fichas1, tArray fichas2, short int &fichasCont, short int fichaNum); //Quita de las fichas del jugador (representadas por ficha1, ficha2 y fichasCont) su ficha fichaNum-ésima (almacenada en las posiciones fichaNum-1 de sendos arrays), desplazando el resto de posiciones a la izquierda para rellenar el hueco dejado por la ficha quitada`
- `bool puedeColocarFicha(const tArray fichas1, const tArray fichas2, short int fichasCont, string tablero); //Comprueba si el jugador puede poner alguna de sus fichas en el tablero`
- `short int sumaPuntos(const tArray fichas1, const tArray fichas2, short int fichasCont); //Suma los puntos acumulados en todas las fichas del jugador`

- Para desordenar el pozo podemos utilizar el algoritmo de Fisher-Yates, que desordena cualquier array de manera no sesgada en un recorrido:

```
void desordenaPozo(tArray pozo1, tArray pozo2) {
   int idx, i;
   short int tmp1, tmp2;
   for (int i = NumFichas - 1; i >= 0; i--) {
      idx = rand() % (i + 1);
      if (i != idx) {
         tmp1 = pozo1[i];
         tmp2 = pozo2[i];
         pozo1[i] = pozo1[idx];
         pozo2[i] = pozo2[idx];
         pozo1[idx] = tmp1;
         pozo2[idx] = tmp2;
      }
   }
}
```