
UNIVERSIDAD
COMPLUTENSE
MADRID

# Fundamentos de la Programación I

Presentación de la Práctica (Versión 1)

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

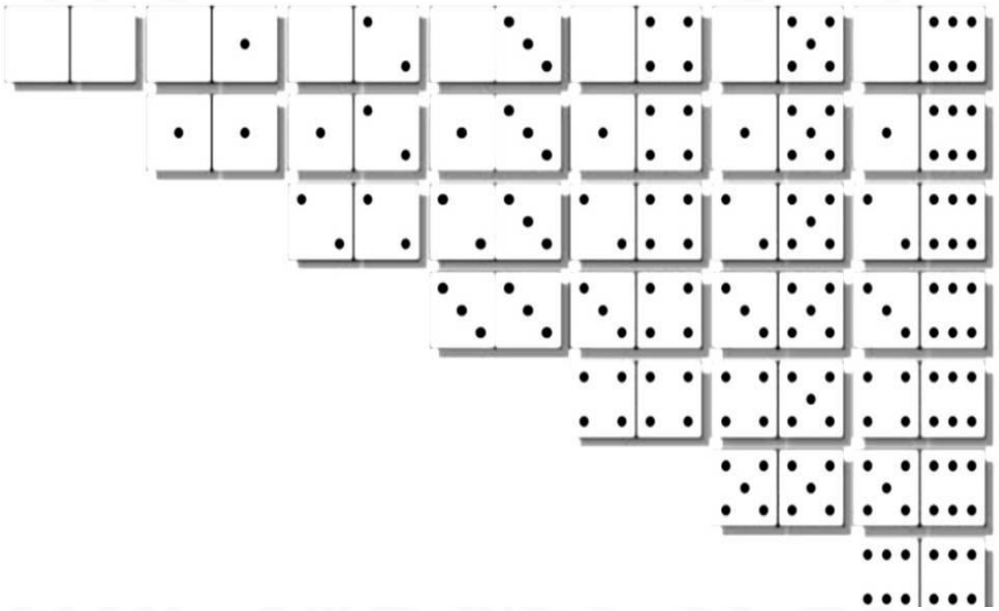
# Índice

# 1. Introducción

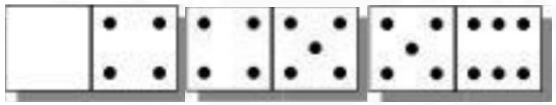
- ✓ En esta práctica se pretende desarrollar, de manera incremental, distintas variantes del juego **dominó**.
- ✓ La práctica cubre los **temas 1 a 5** pasando por **tres versiones**.
- ✓ Algunos **objetivos** concretos son: bucles, subprogramas, entrada/salida, enumerados, arrays, structs sencillos y listas.
- ✓ La fecha de entrega de la **versión 1** es: **10 de noviembre**.
- ✓ La **entrega** debe realizarse a través del campus virtual, utilizando la *tarea de entrega de la práctica 1 (versión 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

- ✓ El dominó es un juego de mesa en el que se emplean unas **fichas** (baldosas) rectangulares, generalmente blancas por la cara y negras por el envés.
- ✓ Una de sus caras está dividida por dos cuadrados, cada uno de los cuales está numerado normalmente mediante disposiciones de puntos como los dados.
- ✓ La **puntuación** habitual es de cero a seis puntos, lo que compone un total de **28 piezas** de dominó siendo la ficha más grande la del seis doble. Existen otras variantes de 55, 91, 136 y 190 piezas.



- ✓ **Reglas**. Cada jugador ha de poner una ficha que cuadre, en uno de sus dos números, con el número de una de las fichas en los extremos de la fila de fichas que haya sobre la mesa (el *tablero*).



- ✓ Si el jugador no puede poner una ficha, debe *robar* fichas de las sobrantes (el *pozo*) hasta que coja una que pueda poner, o, si no quedan, pasar turno.
- ✓ El jugador que gana una ronda, suma los puntos de las fichas de sus adversarios.
- ✓ **Inicio del juego**. Antes de empezar, las fichas se colocan boca abajo sobre la mesa y se mezclan para que los jugadores recojan al azar **7 fichas** de la mesa. Las fichas sobrantes quedan en el **pozo** boca abajo
- ✓ Empieza el jugador que tiene el seis doble. Si ninguno lo tiene empieza el que tiene un cinco doble, y así sucesivamente. Si ningún jugador tiene un doble se ponen todas las fichas en la mesa y se vuelve a empezar.

- ✓ **Desarrollo del juego**. En su turno, cada jugador coloca una de sus fichas en uno de los extremos del tablero (fila de fichas), de forma que uno de sus dos números coincida con el del extremo.
- ✓ Si el jugador no puede poner ficha, debe *robar* fichas del pozo hasta que consiga colocar una. Si no quedan fichas en el pozo, pasa el turno al siguiente jugador.
- ✓ **Final del juego**. Los turnos continúan hasta que se da alguna de las siguientes situaciones:
  - Uno de los jugadores pone su última ficha.
  - Ningún jugador puede poner ficha.

# 3. Funcionalidad de la Aplicación

- ✓ En esta primera versión del juego el programa debe **generar aleatoriamente** una ficha que es colocada inicialmente en la mesa.
- ✓ Tendremos un **único jugador**, que sólo podrá robar una ficha de la mesa, que se generará aleatoriamente.
- ✓ En cada turno, el jugador *roba* una ficha y la intenta colocar junto a la ficha que hay en la mesa, a la izquierda o a la derecha. Podrá salir del juego cuando quiera.

- ✓ En esta versión cada ficha se genera aleatoriamente del conjunto de 28 piezas posibles, sin importar las posibles repeticiones de fichas (cosa que no puede ocurrir en el juego original ni en las siguientes versiones de esta aplicación).
- ✓ Se mantendrá un **contador** de fichas robadas y fichas colocadas.
- ✓ Funciones **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 piezas.

# FP1: Presentación de la Práctica (Versión 1)

```
 ------------------
|     TABLERO      |
 ------------------
|3-2||2-4||4-3||3-6|
Fichas colocadas: 3 - Fichas robadas: 2
Ficha jugador: |1-4|

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

Elija opcion:
```

# 4. Detalles de Implementación

- ✓ A continuación se dan algunas indicaciones para implementar la primera versión.
- ✓ **Generación de números aleatorios**. Los números aleatorios se generan con 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)* (biblioteca *ctime),* ya que así es siempre distinta para cada ejecución. Así pues, el programa deberá ejecutar *srand(time(NULL))* (una sola vez al principio del programa).
- ✓ 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**. Se debe almacenar el tablero en una variable de tipo *string*.
- ✓ Se deben incluir al menos las siguientes **funciones**:
  - `int mostrarMenu() // muestra el menú y devuelve la opción (válida) elegida.`
  - `string fichaToStr(short int izquierda, short int derecha) // devuelve una cadena que representa la ficha: |izquierda-derecha|.`
  - `void mostrarTablero(short int fichaN1, short int fichaN2, string tablero, int numColocadas, int numRobadas) // muestra el tablero, los contadores de fichas robadas y colocadas, y la ficha actual del usuario.`
  - `short int aleat() // devuelve un número aleatorio entre 0 y 6.`
  - `string toStr(int n) // devuelve una cadena que contiene el número.`
  - `bool puedePonerIzq(string tablero, short int fichaN1, short int fichaN2) // indica si se puede colocar la ficha a la izquierda de la fila.`
  - `bool puedePonerDer(string tablero, short int fichaN1, short int fichaN2) // indica si se puede colocar la ficha a la derecha de la fila.`

# FP1: Presentación de la Práctica (Versión 1)

- `string ponerFichaIzq(string tablero, short int fichaN1, short int fichaN2) // coloca la ficha a la izquierda de la fila.`
- `string ponerFichaDer(string tablero, short int fichaN1, short int fichaN2) // coloca la ficha a la derecha de la fila.`