**1. Definición de la clase y variables principales**

internal class Program

{

static List<Stack<int>> torres = new List<Stack<int>>

{

new Stack<int>(),

new Stack<int>(),

new Stack<int>()

};

static int N, movimientos, jugadorActual;

* torres: es una lista con 3 **pilas (Stack<int>)**, cada pila representa una torre.
* N: cantidad de discos.
* movimientos: contador de movimientos de un jugador.
* jugadorActual: sirve para mostrar en pantalla quién está jugando.

**2. Método Main()**

Aquí empieza el juego:

N = PedirEntero("¿Cuántos discos quieres? (3-7): ", 3, 7);

int jugadores = PedirEntero("¿Cuántos jugadores? (1-4): ", 1, 4);

int[] resultados = new int[jugadores];

* Pregunta al usuario cuántos discos usar.
* Pregunta cuántos jugadores habrá.
* Se crea un arreglo resultados para guardar los movimientos de cada jugador.

**2.1. Ciclo para cada jugador**

for (int j = 0; j < jugadores; j++)

{

Inicializar();

movimientos = 0;

jugadorActual = j + 1;

* Para cada jugador:
  + Inicializar() coloca todos los discos en la **torre 1**.
  + Reinicia el contador de movimientos.
  + Guarda quién es el jugador actual.

**2.2. Juego hasta que gane el jugador**

while (torres[2].Count != N && torres[1].Count != N)

{

DibujarTablero();

Console.Write("Movimiento (ej. de torre 1 a torre 3 (1 3), salir=terminar): ");

string entrada = Console.ReadLine();

* El jugador sigue moviendo discos **hasta que todos los discos estén en la torre 2 o en la torre 3**.
* Se dibuja el tablero (DibujarTablero()).
* El jugador escribe un movimiento: por ejemplo 1 3 significa "mover de la torre 1 a la 3".

**2.3. Procesar entrada**

if (entrada == "salir") return;

string[] p = entrada.Split(' ');

int from, to;

if (p.Length == 2 && int.TryParse(p[0], out from) && int.TryParse(p[1], out to))

if (Mover(from - 1, to - 1)) movimientos++;

* Si el jugador escribe salir, termina el programa.
* Divide la entrada en dos números (from, to).
* Si son válidos, llama a Mover().
* Si el movimiento es correcto, aumenta movimientos.

**2.4. Guardar resultado del jugador**

resultados[j] = movimientos;

Console.WriteLine("Jugador " + (j + 1) + " terminó en " + movimientos + " movimientos!");

* Cuando un jugador termina, se guarda cuántos movimientos hizo.

**3. Mostrar resultados finales**

Console.WriteLine("\n=== RESULTADOS ===");

for (int i = 0; i < jugadores; i++)

Console.WriteLine("Jugador " + (i + 1) + ": " + resultados[i] + " movimientos");

int ganador = Array.IndexOf(resultados, resultados.Min());

Console.WriteLine("\n¡Ganador: Jugador " + (ganador + 1) + " con " + resultados[ganador] + " movimientos!");

* Muestra los resultados de todos los jugadores.
* Encuentra el jugador con menos movimientos y lo declara **ganador**.

**4. Métodos auxiliares**

**Inicializar()**

foreach (var t in torres) t.Clear();

for (int d = N; d >= 1; d--) torres[0].Push(d);

* Limpia todas las torres.
* Coloca todos los discos en la torre 1, de mayor a menor (el disco más grande abajo).

**Mover(int from, int to)**

if (from < 0 || from > 2 || to < 0 || to > 2) return false;

if (from == to || torres[from].Count == 0) return false;

if (torres[to].Count > 0 && torres[to].Peek() < torres[from].Peek()) return false;

torres[to].Push(torres[from].Pop());

return true;

* Verifica que el movimiento sea válido:
  + No salir de los límites (torres 0 a 2).
  + No mover de una torre vacía.
  + No poner un disco grande sobre uno más pequeño.
* Si todo está bien, mueve el disco (Pop de la torre origen → Push en la torre destino).

**DibujarTablero()**

Console.Clear();

Console.WriteLine("TORRES DE HANOI\n");

Console.WriteLine("Jugador actual: " + jugadorActual + "\n");

...

* Limpia la pantalla y dibuja las 3 torres.
* Muestra los discos como [1] [2] [3] según el tamaño.
* También enseña cuántos movimientos lleva el jugador.

**Nivel(Stack<int> torre, int nivel)**

int[] arr = torre.Reverse().ToArray();

return nivel < arr.Length ? "[" + arr[nivel] + "]" : " | ";

* Convierte la torre en un arreglo.
* Si en ese nivel hay un disco, lo dibuja con corchetes [ ].
* Si no, muestra un palito | (espacio vacío).

**PedirEntero(string msg, int min, int max)**

int v;

Console.Write(msg);

while (!int.TryParse(Console.ReadLine(), out v) || v < min || v > max)

Console.Write("Valor inválido. " + msg);

return v;

* Pide un número al usuario.
* Repite hasta que sea válido (entre el rango min y max).

**🔑 En resumen:**

Este programa:

1. Pregunta **cuántos discos** y **cuántos jugadores**.
2. Cada jugador juega por turnos el **Torres de Hanoi**.
3. Se valida cada movimiento.
4. Se dibuja el tablero en consola después de cada jugada.
5. Al final, se muestran los resultados y se declara el ganador con menos movimientos.