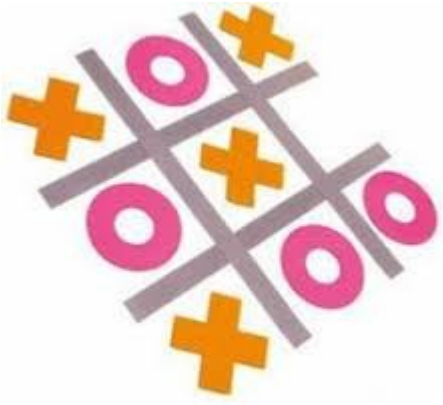
UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Práctica 0. Introducción
El juego del 3 en raya

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada



Estructuras de Datos

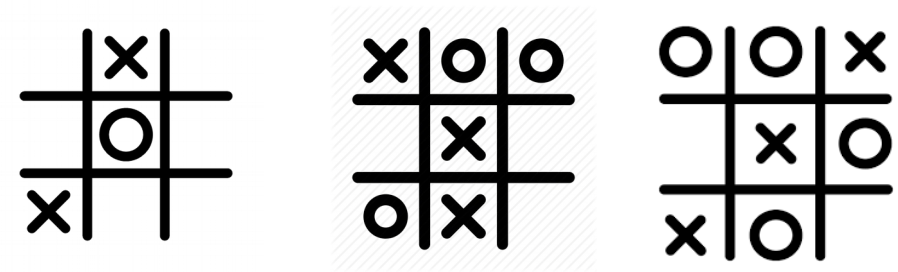
Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

1.- Introducción

Esta práctica inicial solo pretende incidir en la importancia que tiene la abstracción y las estructuras de datos en la resolución de problemas, sirviendo de puente entre las asignaturas de Metodología de la programación II y Estructuras de Datos. Para ilustrarlo todo, se va a poner un ejemplo de un juego: el 3 en raya.

2.- El juego

El tres en raya es un juego de "lapiz y papel" . Tiene numerosas variantes. La más simple es un juego entre dos jugadores (uno de los cuales puede ser un ordenador): O y X, que marcan los espacios de un tablero de 3×3 alternadamente. Cada jugador tiene como objetivo colocar sus fichas en una misma línea recta (horizontal, vertical o diagonal).

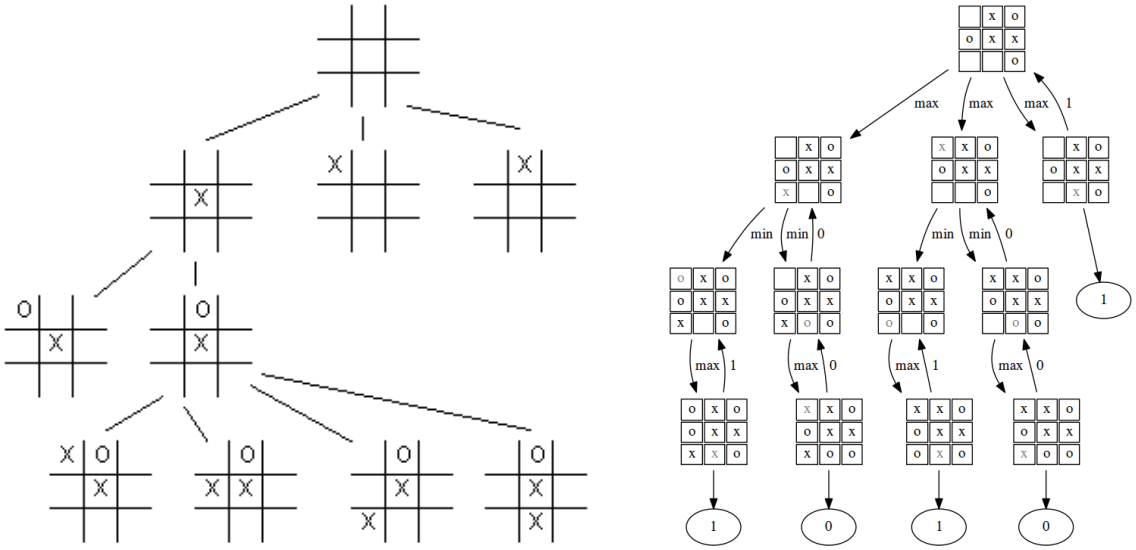


Ejemplos de partidas podrían ser:

*Una partida **ganada por** el primer jugador, **X**:*

*Una partida que termina en **empate**:*

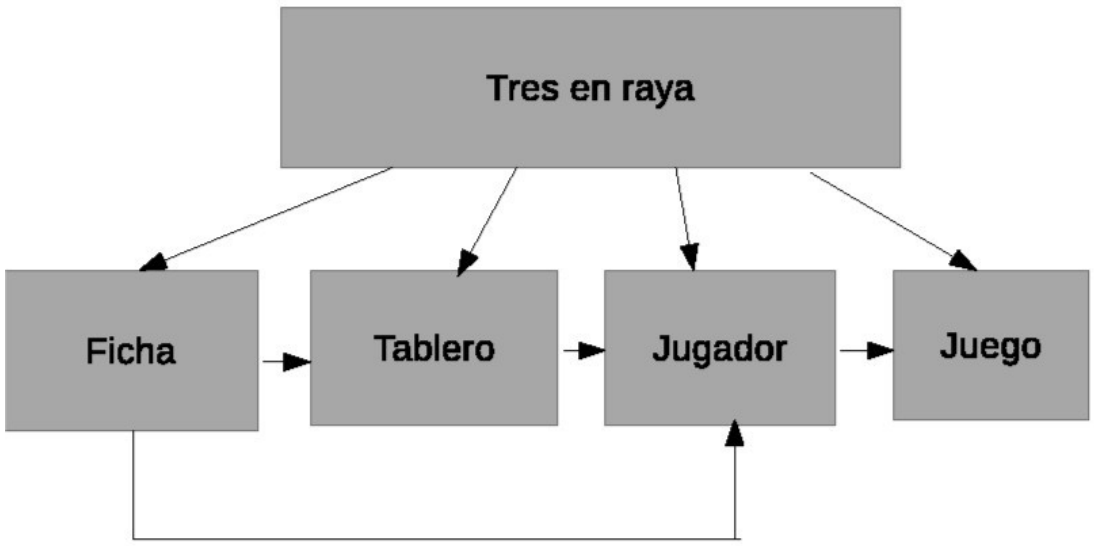
Los jugadores no tardan en descubrir que el juego perfecto termina en empate sin importar qué haga el primer jugador. La simplicidad del juego de tres en raya lo hacen ideal como herramienta pedagógica para enseñar los conceptos de teoría de juegos y como ejemplo básico en la rama de IA que se encarga de la búsqueda de árboles de juegos.



3.- Módulos para la solución

Al iniciar el juego se solicitan los nombres de cada jugador y su nick. En el juego se debe especificar cuál de los jugadores es el primero (por elección o al azar). Puede decidirse que uno de los jugadores sea el ordenador.

Las clases que se implementen en el juego deben tener como base una matriz bidimensional de enteros con un tamaño de 3 por 3 que simulará el tablero del juego. El constructor debe inicializar la matriz : Los jugadores tienen un orden de juego especifico, el primer jugador realiza un movimiento y mostrará una O en el cuadro que desee escoger. Cuando toque al segundo jugador, este realizara un movimiento en el cuadro que desea escoger y mostrará una X. Al realizar cada movimiento el juego debe determinar si se ha ganado o existe un empate. Las clases deben tener atributos, métodos,constructores y las sobrecargas correspondientes. No es necesaria una interfaz gráfica que podría añadirse después.



Definición de ficha:

```
#ifndef __FICHA__H__
#define __FICHA__H__

/* ****************************************** */
#include <iostream>
/* ****************************************** */
// Tipos de fichas que se pueden poner en el tablero
enum Ficha {BLANCO, CIRCULO, CRUZ};
/* ****************************************** */
// Sobrecarga de << para mostrar en ostream el símbolo de la ficha
std::ostream& operator<<(std::ostream &salida, const Ficha &fic);
/* ****************************************** */
#endif
```

Clase tablero:

```
#ifndef __TABLERO__H__
#define __TABLERO__H__
/* ****************************************** */
#include <iostream>
#include "ficha.h"
/* ****************************************** */

class Tablero {

  private:
    Ficha tab[3][3];      // Tablero de fichas 3x3
    int numfichas;        // Número de fichas que han sido puestas en total
    void copia_tablero(const Tablero &orig); // Copia un tablero desde orig
    bool hay3raya(const Ficha &fic) const;// Devuelve true/false si la ficha de tipo fic
                                          //tiene 3 en raya
  public:
    Tablero();                          // Constructor por defecto
    ~Tablero() { };                     // Destructor (vacío)
    Tablero(const Tablero &orig);       // Constructor de copia
    Tablero& operator=(const Tablero &orig);    // Sobrecarga de asignación
    // Inicializa el tablero poniendo en blanco todas sus casillas
    void PonerEnBlanco();     // No es estrictamente necesario

    // Pone una ficha de color fic en la fila f y la columna c
    // Devuelve true si la operación ha tenido éxito y false en caso contrario.
    // Sólo se pueden poner fichas en las casillas que estén en blanco
    bool PonFicha(int f, int c, const Ficha &fic);
    // Devuelve el tipo de ficha que hay en la posición (f,c)
    Ficha QueFichaHay(int f, int c) const;
    // Devuelve si hay algún tipo de ficha que tenga tres en raya. Si no hay 3 en raya
    // devuelve el valor blanco.
    Ficha Busca3Raya() const;
    // Devuelve el número de fichas que hay puestas en el tablero
    int CuantasFichas() const { return numfichas; };
};

// Para mostrar el tablero en pantalla sobrecargamos <<
std::ostream& operator<<(std::ostream &salida, const Tablero &tab);
/* ****************************************** */

#endif
```

Clase jugador:

```
#ifndef __JUGADOR__H__
#define __JUGADOR__H__

/* ****************************************** */
#include <iostream>
#include <string>
#include "ficha.h"
#include "tablero.h"
/* ****************************************** */

class Jugador {
  private:
    std::string nombre;     // Nombre del jugador
    Ficha fic;              // Color de la ficha (cruz o circulo)
    int nivel;              // Nivel del jugador

    // Métodos privados que implementan distintas estrategias de juego
    //   Nivel 0 : Juega una persona
    //   Nivel 1 : Juega la CPU de forma muy básica
    //   ... podríamos implementar nuevos niveles más "inteligentes"
    void piensa_nivel_0(const Tablero &tab, int &fil, int &col) const;
    void piensa_nivel_1(const Tablero &tab, int &fil, int &col) const;

  public:
    // No existe constructor por defecto. Cuando construimos un objeto de tipo
    // jugador debemos asignarle un nombre y un color obligatoriamente.
    Jugador(const std::string &n, const Ficha &f, int ni);

    //    ~Jugador() { };     // El destructor está vacio

    // Obtener el nombre del jugador
    std::string Nombre() const        { return nombre; };

    // Obtener el “color” de la ficha
    Ficha Color() const               { return fic; };

    // Le damos el tablero y nos devuelve dónde quiere poner ficha el jugador
    void PiensaJugada(const Tablero &tab, int &fil, int &col) const;
};

/* ****************************************** */
// Para mostrar los datos del jugador en consola
std::ostream& operator<<(std::ostream &salida, const Jugador &jug);
/* ****************************************** */

#endif
```

Clase juego:

```
#ifndef __JUEGO__H__
#define __JUEGO__H__

/* ****************************************** */
#include "tablero.h"
#include "jugador.h"
/* ****************************************** */

class Juego3Raya {
  private:

    Jugador jug1, jug2;    // Jugadores
    Tablero tab;           // Tablero
    int turno;             // A quien le toca jugar

  public:
    // No existe constructor por defecto
    // Contructor. Para crear un nuevo juego hemos de dar un tablero
    // y dos jugadores obligatoriamente
    Juego3Raya(const Tablero &t, const Jugador &j1, const Jugador &j2);
    ~Juego3Raya()  { };     // Destructor vacío

    void NuevoJuego();      // Prepara el juego para comenzar una nueva partida
    void JugarTurno();      // Avanza un turno

    // Devuelve una referencia (const) al tablero de juego (consultor)
    const Tablero &ElTablero() const { return tab; };

    // Devuelve una referencia al jugador n-ésimo (n=0 ó 1)
    const Jugador &ElJugador(int n) const;

    // Devuelve true si el juego ha terminado (porque haya 3 en raya
    //  o porque haya empate)
    bool HemosAcabado() const;

    // Devuelve el número de jugador a quien le toca poner ficha
    int AQuienLeToca() const  { return turno; };

    // Devuelve el número del jugador que ha ganado. Si aún no ha ganado
    // ninguno o hay empate devuelve -1
    int QuienGana() const;
};

/* ****************************************** */

#endif
```

Cómo se desarrollaría el juego:

```
#include <iostream>
#include <ctime>    // Para función time()
#include <cstdlib>  // Para números aleatorios
#include "ficha.h"
#include "tablero.h"
#include "jugador.h"
#include "juego.h"

using namespace std;

/* ************************************** */

// Preguntamos por teclado los datos de un jugador y lo devolvemos
Jugador LeeJugador(const Ficha f)
{
  string nom;
  int n;
  cout << "Dime el nombre del jugador "<< f <<" : ";
  cin >> nom;
  cout << "  Dime de que nivel es (0=humano, 1=aleatorio)";
  cin >> n;
  return Jugador(nom,f,n);
}

/* ************************************** */

int main(int argc, char *argv[])
{
  char p;

  srand(time(0));  // Inicializamos el generador de números aleatorios
  // Creamos un juego usando un tablero y dos jugadores leidos por teclado
  Juego3Raya juego(Tablero(),LeeJugador(CRUZ),LeeJugador(CIRCULO));

  // También se podría hacer de esta otra forma:
  //   Jugador j1=LeeJugador(cruz);       // Creamos los jugadores
  //   Jugador j2=LeeJugador(circulo);
  //   Tablero tab;                       // Creamos un tablero
  //   Juego3Raya juego(tab,j1,j2);       // Creamos el juego

  do {
    cout << "Los jugadores son: " << endl;
    cout << "  " << juego.ElJugador(0) << endl;
    cout << "  " << juego.ElJugador(1) << endl;
    cout << "Comenzamos!!!" << endl << endl;

    juego.NuevoJuego();    // Comenzamos el juego
    do {
      cout << "Le toca jugar a : " << juego.AQuienLeToca() << endl;
      juego.JugarTurno();    // Avanzamos turno
      cout << "Tras poner la ficha, el tablero queda así: " << endl
           << juego.ElTablero() << endl;
    } while (!juego.HemosAcabado());    // Comprobamos si hemos acabado

    cout << "Se acabó la partida !!!" << endl;

    int ganador=juego.QuienGana();   // Consultamos quien ganó
    if (ganador==-1)
      cout << "Hubo empate" << endl;
    else
      cout << "El ganador ha sido: " << juego.ElJugador(ganador) << endl;

    cout << "¿Otra partida (S/N)?";
    cin >> p;
  } while ((p=='s') || (p=='S'));
}
```