mar 30, 06 13:39 **ficha.h** Page 1/1

```
#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
```

mar 30, 06 13:43 **ficha.cpp** Page 1/1

```
#include <cassert>
#include "ficha.h"

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

std::ostream& operator<<(std::ostream &salida, const Ficha &fic)
{
  assert((fic==BLANCO)||(fic==CIRCULO)||(fic==CRUZ));
  if (fic==BLANCO)
    salida << "  ";
  else if (fic==CIRCULO)
    salida << " O ";
  else   // fic==CRUZ
    salida << " X ";
  return salida;
}

/* ************************************** */
```

```
#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

    // Copia un tablero desde orig
    void copia_tablero(const Tablero &orig);

    // Devuelve true/false si la ficha de tipo fic tiene 3 en raya
    bool hay3raya(const Ficha &fic) const;

  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 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 color de la ficha que hay en la posición (f,c)
    Ficha QueFichaHay(int f, int c) const;

    // Devuelve si hay algún color 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
```



```
#include <cassert>
#include "tablero.h"

using namespace std;

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

Tablero::Tablero()
{
  // El constructor pone en blanco el tablero
  PonerEnBlanco();
}

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

Tablero::Tablero(const Tablero &orig)
{
  copia_tablero(orig);
}

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

void Tablero::PonerEnBlanco()
{
  // Ponemos en blanco el tablero
  for (int i=0; i<3; i++)
    for (int j=0; j<3; j++)
      tab[i][j] = BLANCO;
  numfichas=0;
}

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

Tablero& Tablero::operator=(const Tablero &orig)
{
  if (this!=&orig)
    copia_tablero(orig);
  return *this;
}

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

bool Tablero::PonFicha(int f, int c, const Ficha &fic)
{
  assert((f>=0) && (f<3) && (c>=0) && (c<3));    // Estamos dentro del tablero
  if (tab[f][c]==BLANCO) {                       // No hay ficha en esa casilla
    tab[f][c] = fic;
    numfichas++;
    return true;
  }
  return false;    // Ya hay ficha en esa casilla
}

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

Ficha Tablero::QueFichaHay(int f, int c) const
{
  assert((f>=0) && (f<3) && (c>=0) && (c<3));    // Estamos dentro del tablero
  return tab[f][c];
}
```

```
/* ************************************** */

Ficha Tablero::Busca3Raya() const
{
  if (hay3raya(CIRCULO)) return CIRCULO;
  if (hay3raya(CRUZ)) return CRUZ;
  return BLANCO;     // No hay 3 en raya
}

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

ostream& operator<<(ostream &salida, const Tablero &tab)
{
  salida << " -------------"<<endl;
  for (int i=0; i<3; i++) {
    salida<<" |";
    for (int j=0; j<3; j++)
      salida << tab.QueFichaHay(i,j) << "|";
    salida << endl<<" -------------"<<endl;
  }
  return salida;
}

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

// Método privado
void Tablero::copia_tablero(const Tablero &orig)
{
  for (int i=0; i<3; i++)
    for (int j=0; j<3; j++)
      tab[i][j] = orig.tab[i][j];
  numfichas=orig.numfichas;
}

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

// Método privado
bool Tablero::hay3raya(const Ficha &fic) const
{
  // Buscaremos 3 en raya en las direcciones marcadas por estos vectores
  static const int dir[4][2] = {{1,0},{1,1},{0,1},{1,-1}};

  for (int f=0; f<3; f++) {    //  Recorremos todas las casillas
    for (int c=0; c<3; c++) {  //
      if (tab[f][c]==fic) {        // Cuando encontramos una casilla con fic
        for (int d=0; d<4; d++) {  // Buscamos en las 4 direcciones 3 en raya
          int fx=f, cx=c;          //  a partir de dicha ficha
          fx+=dir[d][0];
          cx+=dir[d][1];
          int numfic=1;
          while ((fx>=0) && (fx<3) && (cx>=0) && (cx<3) && (tab[fx][cx]==fic)) {
            numfic++;
            fx+=dir[d][0];
            cx+=dir[d][1];
          }
          if (numfic==3) return true;
        }
      }
    }
  }
  return false;
}
```



```
#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
```

mar 30, 06 13:48 **jugador.cpp** Page 1/1

```cpp
#include <cassert>
#include <cstdlib>
#include "jugador.h"

using namespace std;

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

Jugador::Jugador(const std::string &n, const Ficha &f, int ni)
                 : nombre(n), fic(f), nivel(ni)
{
  assert((nivel>=0)&&(nivel<2));   // Comprobamos que el nivel es correcto
}

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

std::ostream& operator<<(std::ostream &salida, const Jugador &jug)
{
  salida <<jug.Nombre()<<" ("<<jug.Color()<<")";
  return salida;
}

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

void Jugador::PiensaJugada(const Tablero &tab, int &fil, int &col) const
{
  // En función del nivel del jugador elegimos una estrategia u otra
  switch (nivel) {
    case 0: piensa_nivel_0(tab,fil,col);
            break;
    case 1: piensa_nivel_1(tab,fil,col);
            break;
  }
}

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

void Jugador::piensa_nivel_0(const Tablero &tab, int &fil, int &col) const
{
  cout << " El tablero es: " << endl << tab;
  do {
    cout << " ¿Donde pones ficha (dime fila y columna)? : ";
    cin >> fil >> col;
  } while ((fil<0)||(fil>2)||(col<0)||(col>2));
}

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

void Jugador::piensa_nivel_1(const Tablero &tab, int &fil, int &col) const
{
  cout << " ... estoy pensando ... " << endl;
  do {
    fil = rand()%3;
    col = rand()%3;
  } while (tab.QueFichaHay(fil,col)!=BLANCO);
  cout << "    ... y pongo ficha en (" << fil << "," << col << ")" << endl;
}

/* ************************************** */
```

mar 31, 06 10:32 **juego.h** Page 1/1

```cpp
#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
```

mar 31, 06 10:32 **juego.cpp** Page 1/1

```cpp
#include <cassert>
#include "juego.h"

using namespace std;

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

Juego3Raya::Juego3Raya(const Tablero &t, const Jugador &j1, const Jugador &j2)
            : jug1(j1), jug2(j2), tab(t), turno(0)
{
}

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

void Juego3Raya::NuevoJuego()
{
  turno = (turno+1) % 2;  // Al comenzar un nuevo juego hacemos que comience
                          // a jugar el que perdió en la partida anterior
  tab.PonerEnBlanco();
}

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

void Juego3Raya::JugarTurno()
{
  int f,c;
  Jugador *jug[2] = {&jug1, &jug2};    // Vector de punteros a los jugadores
  // Este vector de punteros se usa para evitar usar un if dentro del bucle

  // Preguntamos al jugador mientras su jugada no sea válida
  do {
    jug[turno]->PiensaJugada(tab,f,c);
  } while (!tab.PonFicha(f,c,jug[turno]->Color()));
  turno = (turno+1) % 2;  // Avanzamos para el siguiente turno
}

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

bool Juego3Raya::HemosAcabado() const
{
  return ((tab.CuantasFichas()==9) || (tab.Busca3Raya()!=BLANCO));
}

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

const Jugador & Juego3Raya::ElJugador(int n) const
{
  assert((n==0)||(n==1));
  return ((n==0) ? jug1 : jug2);
}

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

int Juego3Raya::QuienGana() const
{
  Ficha g = tab.Busca3Raya();
  if (g==jug1.Color())
    return 0;
  else if (g==jug2.Color())
    return 1;
  return -1;    // No gana nadie
}
```

mar 31, 06 10:32 **main.cpp** Page 1/2

```cpp
#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'));
}
```



```
all: raya raya3.ps

juego.o: juego.cpp juego.h tablero.h jugador.h
        g++ -Wall -c juego.cpp

tablero.o: tablero.cpp tablero.h ficha.h
        g++ -Wall -c tablero.cpp

ficha.o: ficha.cpp ficha.h
        g++ -Wall -c ficha.cpp

jugador.o: jugador.cpp jugador.h ficha.h
        g++ -Wall -c jugador.cpp

main.o: main.cpp tablero.h ficha.h jugador.h juego.h
        g++ -Wall -c main.cpp

raya: main.o jugador.o ficha.o tablero.o juego.o
        g++ -o raya main.o jugador.o ficha.o tablero.o juego.o

raya3.ps: main.cpp jugador.cpp ficha.cpp tablero.cpp juego.cpp \
         jugador.h ficha.h tablero.h juego.h Makefile
        a2ps -A fill ficha.h ficha.cpp tablero.h tablero.cpp jugador.h \
                     jugador.cpp juego.h juego.cpp main.cpp \
                     Makefile -o raya3.ps

clean :
        -rm main.o ficha.o jugador.o tablero.o juego.o

mrproper : clean
        -rm raya raya3.ps
```