```
/**
 * @file acumulador.h
 * @author Mario García Márquez y Daniel Alconchel Vázquez
 * @date 13 de Octubre de 2020
 */

#ifndef ACUMULADOR_H
#define ACUMULADOR_H
#include <ostream>

#include "ficha.h"

/**
 * @brief T.D.A. Acumulador
 *
 * Una instancia del tipo acumulador representa el tablero donde vamos a jugar.
 * Se encarga de almacenar las piezas depositadas durante la partida.
 * También almacena la ficha que estamos jugando en un determinado instante de
 * tiempo.
 */

class acumulador {
   private:
    std::vector<std::vector<ficha::tipo>> tablero;

    // Forma de la ficha que esta cayendo, usarse para cambiar tablero cuando
    // la ficha pase a depositada
    ficha::tipo formaCayendo;

   public:
    /**
     * @brief constructor base
     * @note inicializa el tablero a ficha::tipo::vacio
     * @see ficha::tipo
     */
    acumulador();

    /**
     * @brief constructor copia
     * @param orig instancia de la clase acumulador
     */
    acumulador(const acumulador &orig);

    /**
     * @brief constructor con dimensiones
     * @param h altura del tablero
     * @param w anchura del tablero
     * @note inicializa el tablero con esas dimensiones a ficha::tipo::vacio
     * @see ficha::tipo
     */
    acumulador(int h, int w);

    /**
     *  @brief destructor de la clase
     */
    ~acumulador();

    /**
     * @brief imprime el tablero
     * @param os instancia clase ostream
     * @see std::ostream
     * @param ac instancia clase acumulador
     */
    friend std::ostream &operator<<(std::ostream &os, const acumulador &ac);
```

```
    /**
     * @brief coloca la ficha para caer arriba del tablero
     * @param f la ficha que se deja caer
     * @note la coloca en el centro de la parte superior del tablero
     */
    void dejarCaer(ficha f);

    /**
     * @brief indica si la partida ha acabado
     * @retval true si la partida ha acabado(no se puede dejar caer otra ficha)
     */
    bool finDePartida() const;

    /**
     * @brief rota la ficha actual
     * @param horario decide si rotar la ficha en sentido horario o antihorario
     * @note si la ficha no puede ser rotada se mantendrá como está
     */
    void rotarFicha(bool horario);

    /**
     * @brief mueve la ficha actual
     * @param derecha mueve la ficha a la derecha si es true. En caso contrario
     * hacia la izquierda
     * @note si la ficha no puede ser movida se mantendrá como está
     */
    void moverFicha(bool derecha);

    /**
     * @brief deposita la ficha actual
     * @warning usar para bajar la ficha directamente
     */
    void depositarFicha();

    /**
     * @brief determina si la ficha puede seguir cayendo o esta depositada
     * @retval true si la ficha puede seguir cayendo
     * @note en caso de que la ficha este depositada se cambiaran los
     * tipo::activo del tablero por tipo::forma
     */
    bool puedeCaer();

    /**
     * @brief baja la ficha una casilla
     * @warning llamar periodicamente con un reloj, aumentar el reloj para mayor
     * dificultad
     */
    void ticDeReloj();

    /**
     * @brief comprueba si hay filas llenas
     * @retval true si hay filas llenas
     */
    bool filaLlena() const;

    /**
     * @brief borra las filas llenas
     * @return la cantidad de filas eliminadas
     * @note usar con acumulador::filaLlena()
     * @see acumulador::filaLlena()
     */
    int limpiarFilas();
};
```

```
/**
 * @brief imprime el tablero
 * @param os instancia clase ostream
 * @see std::ostream
 * @param ac instancia clase acumulador
 * @warning nota para el profesor, no usamos clase imagen ya que como hemos
 * implementado el acumulador solo hace falta "colorear", es decir, asociar una
 * imagen nxn a cada ficha::tipo
 */
std::ostream &operator<<(std::ostream &os, const acumulador &ac);

#endif
```

```
/**
 * @file cola.h
 * @author Mario García Márquez y Daniel Alconchel Vázquez
 * @date 13 de Octubre de 2020
 */

#ifndef COLA_H
#define COLA_H
#include <ostream>

#include "ficha.h"

/**
 * @brief T.D.A. Cola
 *
 * Una instancia del tipo de dato abstracto cola es un objeto que contiene las 4
 * siguientes fichas a jugar
 */
class cola {
   private:
    /**
     * @brief lista enlazada para representar la cola
     */
    struct elemento {
      ficha valor;
      elemento *siguiente;
    };

    /**
     * @brief representacion interna de la cola como una lista enlazada cerrada
     * @note el ultimo elemento de la lista apunta al primer elemento de esta
     * como el tamaño de la lista siempre es 4 elementos nos aporta una mayor
     * facilidad a la hora de extraer fichas
     */
    elemento *lista;

   public:
    /**
     * @brief constructor de cola
     * @note genera 4 fichas aleatorias y las mete en la cola
     */
    cola();

    /**
     * @brief constructor copia
     * @param orig instancia clase cola
     */
    cola(const cola &orig);

    /**
     * @brief destructor de la clase
     */
    ~cola();

    /**
     * @brief devuelve un vector con las componentes de la lista
     * @return std::vector con todas las componentes de la lista
     */
    std::vector<ficha> getFichas() const;

    /**
     * @brief devuelve el primer elemento de la lista, lo devuelve, lo elimina y
     * añade uno nuevo al final
     * @return la primera ficha de la lista
```

```
     */
    ficha sacarFicha();

    /**
     * @brief imprime el cola
     * @param os instancia clase ostream
     * @see std::ostream
     * @param cl instancia clase cola
     */
    friend std::ostream &operator<<(std::ostream &os, const cola &cl);
};

/**
 * @brief imprime el cola
 * @param os instancia clase ostream
 * @see std::ostream
 * @param cl instancia clase cola
 */
std::ostream &operator<<(std::ostream &os, const cola &cl);

#endif
```

```
/**
 * @file ficha.h
 * @author Mario García Márquez y Daniel Alconchel Vázquez
 * @date 13 de Octubre de 2020
 */

#ifndef FICHA_H
#define FICHA_H
#include <vector>

/**
 * @brief T.D.A. Ficha
 *
 * Una instancia del tipo de dato abstracto Ficha es un objeto que representa
 * una ficha del tetris. Está representado por una forma seleccionada de un
 * enum.
 */
class ficha {
   public:
    /**
     * @brief enum indicando el tipo de ficha
     */
    enum tipo {
      vacio,
      recta,
      cuadrado,
      t,
      lDerecha,
      lIzquierda,
      zDerecha,
      zIzquierda,
      /**
       * @warning Solo usado por acumulador
       */
      activo
    };

    /**
     * @brief constructor base
     */
    ficha();

    /**
     * @brief constructor copia
     * @param orig instancia clase ficha
     */
    ficha(const ficha &orig);

    /**
     * @brief constructor inializando la forma
     * @param forma tipo de la ficha
     * @note se recomienda usar
     */
    ficha(ficha::tipo forma);

    /**
     * @brief destructor de la clase
     */
    ~ficha();

    /**
     * @brief establece la forma de la ficha
     * @param forma la forma a establecer
     */
```

```
    void setForma(ficha::tipo forma);

    /**
     * @brief devuelve la forma establecida
     * @return la forma de la ficha
     */
    tipo getForma() const;

    /**
     * @brief devuelve la matriz asociada a la forma de la ficha
     * @return la matriz asociada a la forma de la ficha
     * @note en la matriz solo se marcan las casillas ocupadas y las no ocupadas
     * tendran el tipo tipo::vacio
     * @see ficha::tipo
     */
    std::vector<std::vector<ficha::tipo>> getMatriz() const;

   private:
    /**
     * @brief forma de la ficha
     */
    tipo forma;
};

#endif
```

```
/**
 * @file juego.h
 * @author Mario García Márquez y Daniel Alconchel Vázquez
 * @date 13 de Octubre de 2020
 */

#ifndef JUEGO_H
#define JUEGO_H
#include "acumulador.h"
#include "cola.h"

class juego {
   private:
    acumulador ac;
    cola cl;
    int piezasJugadas;
    int lineasRotas;
    int dificultad;
    bool jugando;

   public:
    /**
     * @brief imprime el juego
     * @param os instancia clase ostream
     * @see std::ostream
     * @param jg instancia clase juego
     */
    friend std::ostream &operator<<(std::ostream &os, const juego &jg);

    /**
     * @brief metodo alternativo de impresion ya sea porque se desea ventana
     * aparte y no querer usar la salida estandar
     * @warning implementación libre y equivale a la clase Imagen
     */
    void print() const;

    /**
     * @brief constructor base
     */
    juego();

    /**
     * @brief constructor copia
     */
    juego(const juego &orig);

    /**
     * @brief destructor de la clase
     */
    ~juego();

    /**
     * @brief comienza y desarrolla una partida de tetris
     * @warning aqui se incluye toda la lógica del juego
     * @warning finaliza si se fuerza la finalización o si jugando es false(has
     * perdido)
     */
    void jugar();
};

/**
 * @brief imprime el juego
 * @param os instancia clase ostream
 * @see std::ostream
```

```
 * @param jg instancia clase juego
 */
std::ostream &operator<<(std::ostream &os, const juego &jg);

#endif
```