```
/** @author Miguel Cantón Cortés
  * @file Imagen.h
  * @brief Fichero de cabecera asociado a la biblioteca libImagen.a
  *
  * Implementación del TDA imagen (imagen digital de niveles de gris)
  *
  */

#ifndef _IMAGEN_H_
#define _IMAGEN_H_

#include "imagenES.h"

typedef unsigned char byte; ///< Tipo base de cada píxel

/**
  * @brief Clase que almacena la información de una imagen y se encarga de su gestión
  */
class Imagen {
        private:
                int filas; ///< Número de filas de la imagen
                int cols; ///< Número de columnas de la imagen
                byte **img; ///< La imagen en sí: una matriz dinámica 2D de bytes

        public:
        /**
          * @brief Constructor por defecto
          */
                Imagen();
        /**
          * @brief Constructor de copias
          *
          * @param imagen Imagen a copiar
          */
                Imagen(const Imagen &imagen);
        /**
          * @brief Crear una imagen en memoria con fils filas y columnas columnas
          *
          * @param fils filas a reservar
          * @param columnas columnas a reservar
          * @post La imagen creada contiene fils filas y columnas columnas
          */
                Imagen(int fils, int columnas);
        /**
          * @brief Destructor
          */
                ~Imagen();
        /**
          * @brief Operador de asignación
          *
          * @param orig Imagen a copiar
          */
                Imagen& operator= (const Imagen& orig);
        /**
          * @brief Liberar los recursos ocupados por la imagen y crear una imagen en memoria con fils
filas y columnas columnas
          *
          * @param fils filas a reservar
          * @param columnas columnas a reservar
          */
                void Reserva(int fils, int columnas);
        /**
          * @brief Almacena la imagen en un fichero PGM
          *
          * @param salida dirección del archivo a crear
          * @return Si ha tenido éxito en la escritura
          */
                bool guardarPGM(const char *salida) const;
        /**
          * @brief Carga la imagen de un fichero PGM
          *
          * @param salida dirección del archivo a crear
```

```
          * @return Si ha tenido éxito en la carga
          */
                bool cargarPGM(const char *entrada);
        /**
          * @brief Calcular el número de filas de la imagen
          *
          * @return Número de filas de la imagen
          */
                int num_filas() const;
        /**
          * @brief Calcular el número de columnas de la imagen
          *
          * @return Número de columnas de la imagen
          */
                int num_columnas() const;
        /**
          * @brief Asignar el valor valor al píxel (fil, col) de la imagen
          *
          * @param fil fila a modificar
          * @param col columna a modificar
          * @param valor valor a almacenar en (fil, col)
          * @post I(fil,col) == valor
          */
                void asigna_pixel(int fil, int col, byte valor);
        /**
          * @brief Consultar el valor de la casilla (fil, col) de la imagen
          *
          * @param fil fila a consultar
          * @param col columna a consultar
          * @return Valor en (fil, col)
          */
                byte valor_pixel (int fil, int col) const;
};

#endif
```