```cpp
/** @author Miguel Cantón Cortés
  * @file Imagen.cpp
  * @brief Fichero con las definiciones de la clase Imagen asociada a la biblioteca libImagen.a
  *
  * Implementación del TDA imagen (imagen digital de niveles de gris)
  *
  */

#include <cassert>
#include "Imagen.h"

/**
  * @brief Constructor por defecto
  */
Imagen::Imagen() {
        filas = 0;
        cols = 0;
        img = 0;
}

/**
  * @brief Constructor de copias
  *
  * @param imagen Imagen a copiar
  */
Imagen::Imagen(const Imagen &imagen) {
        Imagen(imagen.filas, imagen.cols);
        for(int i = 0; i < filas; i++)
                for(int j = 0; j < cols; j++)
                        img[i][j] = imagen.valor_pixel(i,j);
}

/**
  * @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::Imagen(int fils, int columnas) {
        assert(fils >= 0 && columnas >= 0);
        img = new byte* [fils];
        for(int i = 0; i < fils; i++)
                img[i] = new byte[columnas];
}

/**
  * @brief Destructor
  */
Imagen::~Imagen() {
        for(int i = 0; i < filas; i++)
                delete [] img[i];
        delete [] img;
}

/**
  * @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 Imagen::Reserva(int fils, int columnas) {
        assert(fils >= 0 && columnas >= 0);
        for(int i = 0; i < filas; i++)
                delete [] img[i];
        delete [] img;
        img = new byte* [fils];
        for(int i = 0; i < fils; i++)
                img[i] = new byte[columnas];
        filas = fils;
        cols = columnas;
```

```
}

/**
  * @brief Operador de asignación
  *
  * @param orig Imagen a copiar
  */
Imagen& Imagen::operator= (const Imagen& orig) {
        assert(&orig != this);
        Reserva(orig.filas, orig.cols);
        for(int i = 0; i < filas; i++)
                for(int j = 0; j < cols; j++)
                        img[i][j] = orig.valor_pixel(i,j);
        return *this;
}

/**
  * @brief Almacena la imagen en un fichero PGM
  *
  * @param salida dirección del archivo a crear
  * @return Si ha tenido éxito en la escritura
  */
bool Imagen::guardarPGM(const char *salida) const {
        byte *imagen_S = new byte[filas*cols];
        bool correcto;

        //Pasamos a vector la matriz de la imagen
        for(int i = 0; i < filas; i++)
                for(int j = 0; j < cols; j++)
                        imagen_S[i*cols+j] = img[i][j];

        correcto = EscribirImagenPGM(salida, imagen_S, filas, cols);
        delete [] imagen_S;
        return correcto;
}

/**
  * @brief Carga la imagen de un fichero PGM
  *
  * @param salida dirección del archivo a crear
  * @return Si ha tenido éxito en la carga
  */
bool Imagen::cargarPGM(const char *entrada) {
        int fils, columnas;
        byte *imagen_E = 0;
        if(IMG_PGM == LeerTipoImagen(entrada)) {
                imagen_E = LeerImagenPGM(entrada, fils, columnas);
                Reserva(fils, columnas);

                //Pasamos el vector a la matriz de la imagen
                for(int i = 0; i < filas; i++)
                        for(int j = 0; j < cols; j++)
                                asigna_pixel(i, j, imagen_E[i*cols+j]);
                delete [] imagen_E;
        }
        else
                return false;

        return true;
}

/**
  * @brief Calcular el número de filas de la imagen
  *
  * @return Número de filas de la imagen
  */
int Imagen::num_filas() const { return filas; }

/**
  * @brief Calcular el número de columnas de la imagen
  *
  * @return Número de columnas de la imagen
```

```
  */
int Imagen::num_columnas() const { return cols; }

/**
  * @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 Imagen::asigna_pixel(int fil, int col, byte valor) {
        assert(0 <= fil); assert(fil < filas); assert(0 <= col); assert(col < cols); assert(0 <=
valor); assert(valor <= 255);
        img[fil][col] = 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 Imagen::valor_pixel(int fil, int col) const { return img[fil][col]; }
```