Práctica 2: Abstracción. TDA Imagen

Estructuras de Datos - Curso 21/22

Alumnos:

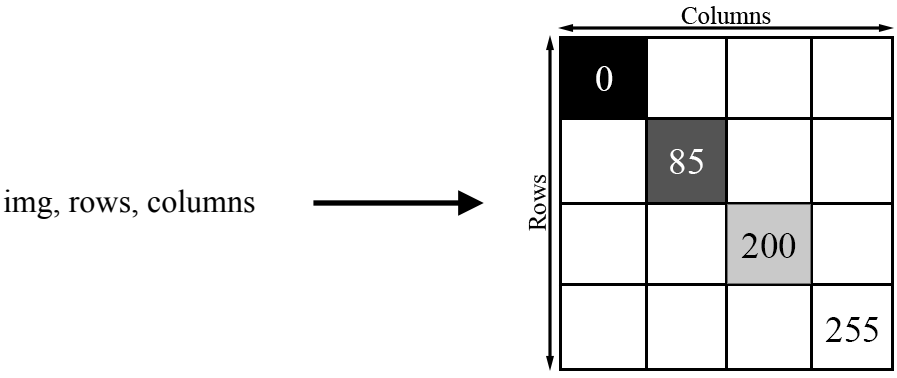
- Feixas Galdeano, José Miguel.
- Piqueras Brück, Andrés.

I. Breve introducción a la práctica

En esta segunda práctica de la asignatura de Estructuras de Datos, la primera puntuable, se nos plantea el desarrollo de un tipo de dato abstracto que nos permita trabajar con imágenes en escala de grises, a tal efecto nos fueron proporcionados los archivos imagenES.h e imagenES.cpp, así como una colección de imágenes en formato PGM para poder leer y escribir archivos en el formato mencionado. En cuanto a nuestra labor en esta práctica encontramos dos partes claramente diferenciadas, la primera se trata del desarrollo completo del tipo de dato abstracto imagen, desde los primeros procesos de conceptualización y abstracción hasta la depuración del mismo, pasando como no podía ser de otra manera por la implementación, etapas que tendremos a bien repasar más a fondo en sucesivos apartados de este documento. En segundo lugar se nos propuso el desarrollo de una serie de funciones para la edición y procesamiento de imágenes a fin de trabajar con la clase imagen desarrollada, así como un *main* donde poder usar estas herramientas.

II. El tipo de dato abstracto imagen

El desarrollo del tipo de dato abstracto imagen fue la primera tarea que realizamos para esta práctica, comenzamos este proceso siendo el proceso de abstracción que venía recogido en el propio guion de la práctica, el cual consideramos más que adecuado para el desarrollo de la práctica. Por tanto, nuestro tipo de dato abstracto imagen sería implementado en la clase *image*, siendo el tipo *rep* de este TDA una matriz bidimensional dinámica, llamada *img*, de tipo *byte*, un tipo previamente definido como un *unsigned char* nativo de C++, los cuales huelga decir que varían exclusivamente entre los valores 0 y 255. Cada celda de esta matriz contendrá un valor de dicho tipo que representará el nivel de gris de ese *píxel*, a fin de representar una imagen en escala de grises. Además también nos encontraremos con dos parámetros de tipo entero, *rows* y *columns*, que representan respectivamente el número de filas y columnas de nuestra matriz.



Función de abstracción del TDA imagen

Los objetos de la clase *image* desarrollada deben de cumplir la condición de que tanto *rows* como *columns* han de ser mayores o iguales que cero para que se consideren objetos válidos en nuestra implementación, está condición constituye el invariante de representación, mediante el cual podemos asegurar que aquellos objetos que cumplan este se podrán decir objetos del tipo abstracto imagen.

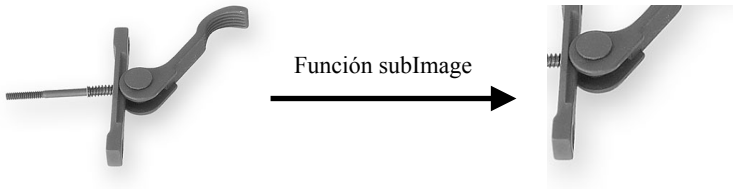
En cuanto a la implementación de la clase *image*, se realizaron todas las funciones que se indicaban en el guion de la práctica, como los constructores, el operador de asignación y algunas que implementaban funcionalidades básicas de cualquier clase, como los *getters and setters*. Además, durante el transcurso de la práctica se vio conveniente la implementación de otras funciones, tanto publicas como privadas, para el correcto desarrollo de la misma. Concretamente se decidió implementar las más que usuales funciones de reserva y liberación de la memoria dinámica, una función de copia para evitar la repetición de código en varias funciones que requieren de esta funcionalidad y otra para redimensionar una imagen. Además también se añadieron otros métodos fruto de la implementación de la segunda parte de la práctica, como pueden ser los de añadir una fila o una columna o los de realizar la interpolación bilineal.

III. Funciones de procesamiento de imágenes

En esta segunda parte de la práctica desarrollamos algunos de los métodos recogidos en el guion para el procesamiento de imágenes, concretamente se eligió realizar la función uno, «extracción de subimagénes», y la tres, «zoom de una porción de la imagen», además de la cinco, «aumento de contraste de una imagen mediante el estiramiento del histograma», que era obligatoria. Estas quedan recogidas e implementadas en los archivos tools.h y tools.cpp, donde además figuran otras funciones que se consideraron beneficiosas para el desarrollo del código, fundamentalmente se trata de métodos que nos permiten pasar de formato PGM a nuestro tipo *image* y guardar las imágenes en el mencionado formato.

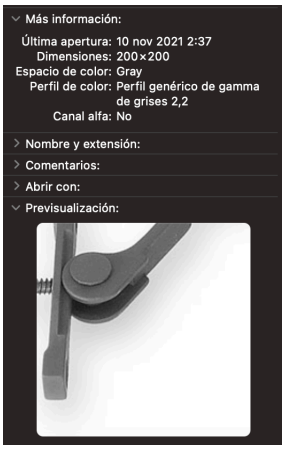
Ejercicio 1. Extracción de subimágenes.

En este caso nos encontramos con una funcionalidad que resultó sencilla de implementar, la mayor dificultad residió en poder controlar correctamente la copia desde la imagen original a la nueva imagen recortada, teniendo en cuenta la más que probable posibilidad de que pudiésemos llegar a los extremos de la imagen original en algunos casos.



Ejemplo de ejecución de la función para extraer una subimagen

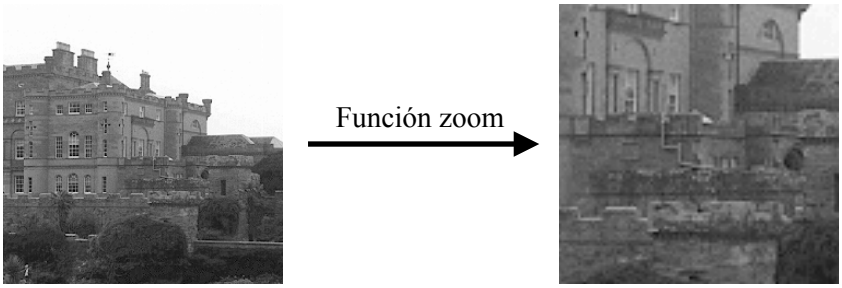
Para llevar a cabo este ejemplo se usó la imagen llave.pgm, se indicó comenzar desde el píxel (75,150) de la misma y a la subimagen se le dio un tamaño de 200 por 200, el cual podemos ver que es correcto comprobando los metadatos de la imagen en el propio explorador del sistema operativo.



Metadatos de la subimagen obtenida

Ejercicio 3. Zoom de una porción de la imagen.

Esta se trató de la función más laboriosa a la hora de llevar a cabo su implementación dado que contenía una serie de procedimientos para los que decidimos crear métodos en la clase *image* a fin de evitar la repetición de código. Además del ejemplo de funcionamiento se adjunta a modo de curiosidad dos resultados de ejecución de la función zoom con los mismos parámetros y sobre la misma imagen, con la diferencia de que en el primero se replican los píxeles y en el segundo se realiza la interpolación bilineal, a simple vista se puede apreciar la enorme diferencia de calidad que existe entre los dos métodos.



Ejemplo de ejecución de la función zoom

Zoom obtenido replicando píxeles

Zoom obtenido por interpolación bilineal

Ejercicio 5. Aumento de contraste de una imagen mediante un estiramiento del histograma.

El desarrollo de este método tenía como pilar fundamental el comprender correctamente las transformaciones matemáticas que se debían implementar sobre los datos de la matriz para que el aumento de contraste se haga de forma correcta.

Ejemplo de ejecución de la función de estiramiento del histograma